```
classdef class_serial_port < handle
    %CLASS_SERIAL_PORT Communication with hardware via serial port
    %
    properties (SetAccess = immutable)
        port            = '';
        baudrate        = '';
        terminator      = '';
        terminator_text = '';
        battery_log     = false;

        callback_receive = false;
    end

    properties (Access = private)
        handle = false;
    end

    methods
        % constructor
        function obj = class_serial_port(port, baudrate, terminator, callback_receive)
            obj.port = port;
            obj.baudrate = baudrate;
            obj.terminator = terminator;
            obj.callback_receive = callback_receive;

            switch terminator
                case 'LF'
                    obj.terminator_text = '\n';
                case 'CR'
                    obj.terminator_text = '\r';
                case 'CR/LF'
                    obj.terminator_text = '\r\n';
                case 'LF/CR'
                    obj.terminator_text = '\n\r';
                otherwise
                    error('Unknown COM-PORT terminator string %s', terminator);
            end

            filename = datestr(now,'yymmdd_HHMMSS');
            obj.battery_log = fopen(sprintf('battery_log_%s.log', filename),'w');
        end

        function success = connect(obj)
            try
                % Find a serial port object.
```

```
46                  obj.handle = instrfind('Type', 'serial',
                    'Port', obj.port, 'Tag', '');
47
48                  % Create the serial port object if it does not
                    exist
49                  % otherwise use the object that was found.
50                  if isempty(obj.handle)
51                      obj.handle = serial(obj.port);
52                  end
53
54                  % Configure instrument object, comport.
55                  set(obj.handle, 'BaudRate', obj.baudrate);
56                  set(obj.handle, 'Terminator', {obj.terminator,
                    obj.terminator});
57                  obj.handle.BytesAvailableFcnMode = 'terminator';
58                  obj.handle.BytesAvailableFcn = @obj.loop;
59
60                  % Connect to instrument object, comport.
61                  fopen(obj.handle);
62
63                  if obj.isOpen
64                      fprintf('Connected to COM-Port %s.\n',
                        obj.port);
65                      success = true;
66                  else
67                      warning('Connecting to COM-Port %s
                        failed.', obj.port);
68                      success = false;
69                  end
70              catch e
71                  warning('Connecting to COM-Port %s failed: %s',
                    obj.port, getReport(e));
72                  success = false;
73              end
74          end
75
76          function close(obj)
77              if obj.handle ~= false
78                  obj.setDemoMode(0);
79                  obj.setLed(0);
80                  obj.setHalogen(0);
81                  obj.setTrainSpeed(0);
82
83                  if obj.battery_log ~= -1
84                      fclose(obj.battery_log);
85                  end
86
87                  fclose(obj.handle);
88                  delete(obj.handle);
```

```
89              end
90
91              clear obj.handle;
92
93              obj.handle = false;
94
95              fprintf('Closed connection to COM-Port %s.\n',
                obj.port);
96          end
97
98          function open = isOpen(obj)
99              open = obj.handle ~= false && isvalid(obj.handle)
                && strcmp(get(obj.handle, 'Status'), 'open');
100         end
101
102         function success = send(obj, text)
103             if obj.isOpen
104                 fprintf(obj.handle, sprintf('%s%s', text,
                    obj.terminator_text));
105                 fprintf('SerialPort write: "%s"\n', text);
106                 pause(0.05);
107                 success = true;
108             else
109                 warning('COM-Port %s is not connected, can''t
                    send data.', obj.port);
110                 success = false;
111             end
112         end
113
114         function success = setLed(obj, state)
115             if(state < 0 || state > 4)
116                 error('led state range is 0 - 4');
117             end
118
119             fprintf('Set LED state to %d.\n', state);
120
121             success = obj.send(sprintf('&L:%d;', state));
122         end
123
124         function success = setHalogen(obj, state)
125             if(state ~= 0 && state ~= 1 && state ~= 4)
126                 error('halogen states are 0, 1 and 4');
127             end
128
129             fprintf('Set Halogen state to %d.\n', state);
130
131             success = obj.send(sprintf('&H:%d;', state));
132         end
133
```

```
134         function success = setDemoMode(obj, state)
135             if(state ~= 0 && state ~= 1)
136                 error('demo mode states are 0 and 1');
137             end
138 
139             fprintf('Set DemoMode state to %d.\n', state);
140 
141             success = obj.send(sprintf('&d:%d;', state));
142         end
143 
144         function success = setTrainSpeed(obj, speed, left)
145             if nargin < 3
146                 left = true;
147             end
148 
149             if left == true
150                 left = 1;
151             else
152                 left = 0;
153             end
154 
155             if speed < 0 || speed > 10
156                 error('train speed range is 0 - 10');
157             end
158 
159             fprintf('Set train speed to %d and direction to
                %d.\n', speed, left);
160 
161             success = obj.send(sprintf('&D;%d;%d;', left,
                speed));
162         end
163 
164         function loop(obj, handle, ~, ~)
165             try
166                 while (handle.BytesAvailable > 0)
167                     line = fgetl(handle);
168                     fprintf('SerialPort read: "%s"\n', line);
169 
170                     switch line
171                         case 'PreLap Sensor 1'
172                             % (trigger capture for triangulation)
173                             obj.call_callback('prelap1', false);
174                         case 'Lap Sensor 1'
175                             % end of round 1
176                             obj.call_callback('lap1', false);
177                         case '...Slow'
178                             % (trigger triangulation)
179                             obj.call_callback('prelap0', false);
180                         case '...Stop'
```

```
                            % end of round 2
                            % (trigger QR-Code, infrared and
                            multispectral)
                            obj.call_callback('lap0', false);
                        case 'Set Halo 1'
                            % end of round 2
                            % (trigger QR-Code, infrared and
                            multispectral)
                            obj.call_callback('halo', false);
                        case 'Set LED LED1+2'
                            % end of round 2
                            % (trigger QR-Code, infrared and
                            multispectral)
                            obj.call_callback('led', false);
                    end

                    if strncmp(line, 'BAT:', 4)
                        % log battery state
                        if obj.battery_log ~= -1
                            fprintf(obj.battery_log, '%s
                            %s\r\n',
                            datestr(now,'yymmdd_HHMMSS'),
                            line(5:end));
                        end

                        % show battery state
                        obj.call_callback('bat', line(5:end));
                    end
                end
            catch e
                warning('SerialPort read error: %s',
                getReport(e));
            end
        end

        function call_callback(obj, type, parameter)
            if isa(obj.callback_receive, 'function_handle')
                obj.callback_receive(type, parameter);
            elseif iscell(obj.callback_receive) &&
            isa(obj.callback_receive{1}, 'function_handle')
                obj.callback_receive{1}(type, parameter,
                obj.callback_receive{2:end});
            else
                error('SerialPort callback is not callable');
            end
        end

        function delete(obj)
            obj.close();
```

```
221             end
222         end
223
224     end
225
```