```
 1: program robuster_Ansatz;
 2:
 3: {$ifdef MSWINDOWS}{$apptype CONSOLE}{$endif}
 4: {$ifdef FPC}{$mode OBJFPC}{$H+}{$endif}
 5:
 6: uses
 7:  {$IFDEF UNIX} cthreads, {$ENDIF}
 8:   SysUtils,
 9:   IPConnection,
10:   Device,
11:   BrickletTemperatureV2,
12:   BrickletLCD128x64;
13:
14: type
15:   TExampleRugged = class
16:   private
17:     ipcon: TIPConnection;
18:     TEMP_V2: TBrickletTemperatureV2;
19:     LCD_128x64: TBrickletLCD128x64;
20:   public
21:     procedure TemperatureCB(Sender: TBrickletTemperatureV2;
22:       const temperature: smallint);
23:     procedure EnumerateCB(Sender: TIPConnection; const uid: string;
24:       const connectedUid: string; const position: char;
25:       const hardwareVersion: TVersionNumber;
26:       const firmwareVersion: TVersionNumber;
27:       const deviceIdentifier: word; const enumerationType: byte);
28:     procedure Execute;
29:   end;
30:
31: const
32:   HOST = 'localhost';
33:   PORT = 4223;
34:   tempUID = 'KEd';
35:   LCD_128x64_UID = '21q9';
36:
37: var
38:   e: TExampleRugged;
39:
40:   {Callback procedure for temperature and output to LCD_128_64 }
41:   procedure TExampleRugged.TemperatureCB(Sender: TBrickletTemperatureV2;
42:   const temperature: smallint);
43:
44:   begin
45:     Writeln(Format('Temperature: %f°C', [temperature/100.0]));
46:     if Assigned(LCD_128x64) then
47:     begin
48:       LCD_128x64.ClearDisplay;
49:       LCD_128x64.WriteLine(0, 0,
50:         Format('Temperature: %f°C', [temperature / 100.0]));
51:     end;
52:   end;
53:
54:   procedure TExampleRugged.EnumerateCB(Sender: TIPConnection;
55:   const uid: string; const connectedUid: string;
56:   const position: char; const hardwareVersion: TVersionNumber;
57:   const firmwareVersion: TVersionNumber; const deviceIdentifier: word;
58:   const enumerationType: byte);
59:   begin
```

```
60:     { Print incoming enumeration }
61:
62:     WriteLn(format('UID:               %s', [uid]));
63:     WriteLn(Format('Enumerate Type:    %d', [enumerationType]));
64:
65:     if (enumerationType = IPCON_ENUMERATION_TYPE_CONNECTED) or
66:       (enumerationType = IPCON_ENUMERATION_TYPE_AVAILABLE)
67:     { or (enumerationType <> IPCON_ENUMERATION_TYPE_DISCONNECTED) } then
68:     begin
69:       { Print incoming enumeration }
70:
71:       WriteLn(Format('Connected UID:     %s', [connectedUid]));
72:       WriteLn(Format('Position:          %s', [position]));
73:       WriteLn('Hardware Version:  ',
74:         Format('%d', [hardwareVersion[0]]),
75:         Format('.%d', [hardwareVersion[1]]),
76:         Format('.%d', [hardwareVersion[2]]));
77:       WriteLn('Firmware Version:  ',
78:         Format('%d', [firmwareVersion[0]]),
79:         Format('.%d', [firmwareVersion[1]]),
80:         Format('.%d', [firmwareVersion[2]]));
81:       WriteLn(Format('Device Identifier: %d', [deviceIdentifier]));
82:       WriteLn;
83:
84:       if deviceIdentifier = Bricklet_temperature_v2_device_identifier then
85:       begin
86:         if not Assigned(Temp_V2) then
87:         begin
88:           { Create device Object }
89:           WriteLn('TEMP_V2: Create device Object');
90:           TEMP_V2 := TBrickletTemperatureV2.Create(tempUID, ipcon);
91:           { Register temperature callback to proceduer TemperatureCB }
92:           TEMP_V2.OnTemperature := {$ifdef FPC}@{$endif}TemperatureCB;
93:         end;
94:         { set period for temperature callback to 1s without teshold }
95:         TEMP_V2.SetTemperatureCallbackConfiguration(1000, False, 'x', 0, 0);
96:       end;
97:
98:
99:
100:      if deviceIdentifier = BRICKLET_LCD_128X64_DEVICE_IDENTIFIER then
101:       begin
102:         if not Assigned(LCD_128x64) then
103:         begin
104:           WriteLn('LCD_128x64: Create device Object');
105:           LCD_128x64.Create(LCD_128x64_UID, ipcon);
106:
107:           if Assigned(LCD_128x64) then
108:             WriteLn('LCD_128x64 created');
109:         end;
110:       end;
111:
112:
113:
114:
115:     end;
116:   end;
117:
118:   procedure TExampleRugged.Execute;
```

```
119:   begin
120:     { Create connection }
121:     ipcon := TIPConnection.Create;
122:
123:     { Register enumerate callback to "EnumerateCB" }
124:     ipcon.OnEnumerate := {$ifdef FPC}@{$endif}EnumerateCB;
125:
126:     { Connect to brickd }
127:     ipcon.Connect(HOST, PORT);
128:
129:     { Don't use device before ipcon is connected }
130:
131:     { Trigger enumerate }
132:     ipcon.Enumerate;
133:
134:     WriteLn('Press key to exit');
135:     ReadLn; //Wartet auf Eingabe waehrendessen callback Ausgaben.
136:
137:     { Calls LCD_128x64.Free internally }
138:     FreeAndNil(LCD_128x64);
139:     { Calls TEMP_V2.Free internally }
140:     FreeAndNil(TEMP_V2);
141:
142:     { Calls ipcon.Free internally }
143:     FreeAndNil(ipcon);
144:   end;
145:
146: begin
147:   e := TExampleRugged.Create;
148:   e.Execute;
149:   e.Free;
150: end.
```