```
// UserFiniteStateMachine.h

//=============================================================================================
//  Application code
//
//  Name:    AutoAlarm
//  Version: 1.0
//  Date:    28-5-2017
//  Author:  Jelle Siemonsma
//
//  Short description:
//
//  Car alarm.
//
//  Used hardware components: MEGA2560, SIM900 shield, a prepaid SIM card, NEO-6M GPS.
//  Used software: the standard AFSM machine.
//
//  The ideal situation is a permanent power supply in the car of 12VDC, but the system
//  also handles a 12VDC switched supply. When the car is driving, it's moves are
//  monitored by the GPS; uses velocity and location. When the system is connected to a
//  switched 12VDC, the GPS has to make a cold start and an accurate signal will be
//  available -depending on the situation- in about 3 minutes from cold start.
//  Every 30 minutes driving, a SMS is send to the given phone number. After stopping/
//  parking, another SMS is send to the phone in the format of google maps. Opening
//  the SMS will give the hybrid presentation of the location. Sending "TESTSMS" and
//  "RESETTEST" to the shield, will also give a SMS with the current location. Sending
//  "SALDO" to the shield, will give another SMS with the current amount of credit on
//  the shield.
//
//  www.jbsiemonsma.nl
//
//*============================================================================================

String DegreeLat(String DegreeString)
{
```

```
36      float Degree;
37      float Min;
38      float LongLat;
39
40      Degree = DegreeString.substring(0, 2).toFloat();
41      Min = DegreeString.substring(2).toFloat();
42      LongLat = Min / 60;
43      LongLat = Degree + LongLat ;
44
45      return String(LongLat,6);
46
47  }
48
49  String DegreeLong(String DegreeString)
50  {
51      float Degree;
52      float Min;
53      float LongLat;
54
55      Degree = DegreeString.substring(0, 3).toFloat();
56      Min = DegreeString.substring(3).toFloat();
57      LongLat = Min / 60;
58      LongLat = Degree + LongLat ;
59
60      return String(LongLat, 6);
61
62  }
63
64
65  void UserFSM()
66  {
67      // ================================================================================================================
68      // START
69      //
70      // Machine state "START" is the default state at startup of the board!
```

```
71      // Starting the pahse for sending a test SMS message end the phase for waiting on a good connection
72      // with the GPS; no. of sats bigger than 2 and horizontal accuracy is better then 50m.
73      // ================================================================================================================
74      //
75      if (MachineState == FiniteState("START"))
76      {
77          //PHASE ACTIONS
78          //TRANSITION CONDITIONS//
79          TransitionToState("TEST");
80          TransitionToState("WAITING");
81      };
82
83      // ================================================================================================================
84      // WAITING
85      //
86      // Wating on a good signal from the GPS device. When ok, check the velocity. When velocity is higher then 10km/h
87      // the car is probably moving, goto the "PENDINGDRIVE" state otherwise goto the "PENDINGSTOP" state, the car is
88      // probably not moving.
89      // ================================================================================================================
90      //
91      if (MachineState == FiniteState("WAITING"))
92      {
93          if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
94          {
95            if (GPSdata.SOG.toFloat() > 10.0) TransitionToState("PENDINGDRIVE"); else TransitionToState("PENDINGSTOP");
96            //SendSMS("The GPS is accurate and following the car.");
97          }
98      };
99
100     // ================================================================================================================
101     // PENDINGDRIVE
102     //
103     // The car is probably moving. When the car is moving for at least 15s, the car is driving; goto state "DRIVING".
104     // Check the velocity of the car, when moving slower then 10km/h, presume the car is not moving; goto state "NOTMOVING".
105     // ================================================================================================================
```

```
106     //
107     if (MachineState == FiniteState("PENDINGDRIVE"))
108     {
109         if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
110         {
111             if (Timer(15000, TimPendingDrive))
112             {
113                 TransitionToState("DRIVING");
114                 Stopped = false;
115             }
116             else
117                 if (GPSdata.SOG.toFloat() < 10.0)
118                 {
119                     CancelTimer(TimPendingDrive);
120                     TransitionToState("NOTMOVING");
121                 }
122         }
123         else TransitionToState("WAITING");
124 
125     }
126 
127     // ================================================================================================================
128     // PENDINGSTOP
129     //
130     // The car is probably stopped. When the car is not moving for 15s, the car is stopped; goto state "STOPPED".
131     // Check the velocity of the car, when going faster then 10km/h, presume the car is still moving; goto state "DRIVING".
132     // ================================================================================================================
133     //
134     if (MachineState == FiniteState("PENDINGSTOP"))
135     {
136         if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
137         {
138             if (Timer(15000, TimPendingStop)) TransitionToState("NOTMOVING");
139             else
140                 if (GPSdata.SOG.toFloat() > 10.0)
```

```
141                 {
142                     CancelTimer(TimPendingStop);
143                     TransitionToState("DRIVING");
144                 }
145 
146         }
147         else TransitionToState("WAITING");
148     }
149 
150     // ================================================================================================================
151     // NOTMOVING
152     //
153     // The car is not moving. When moving faster then 10km/h, persume the car is probably driving; goto state "PENDINGRIVE".
154     // Send a text message (SMS) entering the state, send the cars position (one time).
155     // ================================================================================================================
156     //
157     if (MachineState == FiniteState("NOTMOVING"))
158     {
159         if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
160         {
161             if (GPSdata.SOG.toFloat() > 10.0) TransitionToState("PENDINGDRIVE");
162             else
163             {
164                 if (!Stopped) SendSMS("https://maps.google.com?t=h&q=loc:" + DegreeLat(GPSdata.Latitude) + "+" + DegreeLong
                      (GPSdata.Longitude));
165                 Stopped = true;
166             }
167         }
168         else TransitionToState("WAITING");
169     }
170 
171     // ================================================================================================================
172     // DRIVING
173     //
174     // The car is driving. When the car is driving slower then 10km/h, the car is probably pending to stop; goto
```

```
175     // state "PENDINGSTOP".
176     // Send every 30min a message with the position of the car and the actual speed.
177     // ================================================================================================================
178     //
179     if (MachineState == FiniteState("DRIVING"))
180     {
181         if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
182         {
183             if (GPSdata.SOG.toFloat() < 10.0)
184             {
185                 TransitionToState("PENDINGSTOP");
186                 CancelTimer(TimInterval);
187             };
188             if (Timer(1800000, TimInterval)) SendSMS("https://maps.google.com?t=h&q=loc:" + DegreeLat(GPSdata.Latitude) + "+" ↵
                    + DegreeLong(GPSdata.Longitude));
189         }
190         else TransitionToState("WAITING");
191     }
192
193
194     // ================================================================================================================
195     // TEST
196     //
197     // On the rising edge of input "TESTSMS" on the 22e pin, send a test SMS with the position of the car and the
198     // actual speed. You may also want to send an SMS with text "TEST" and reset with SMS text "RESET" for the
199     // same result as toggling pin 22.
200     // ================================================================================================================
201     //
202     if (MachineState == FiniteState("TEST"))
203     {
204         if (DigRising("TESTSMS"))
205         {
206             if ((GPSdata.GU.toInt() > 2) && (GPSdata.Hacc.toFloat() < 50.0))
207                 SendSMS("https://maps.google.com?t=h&q=loc:" + DegreeLat(GPSdata.Latitude) + "+" + DegreeLong ↵
                      (GPSdata.Longitude));
```

```
208             else
209                 SendSMS("GPS not accurate enough, no. sats=" + String(GPSdata.GU.toInt()) + ", Horizontal acc.=" + String
                      (GPSdata.Hacc.toFloat(), 2));
210         }
211     }
212
213
214
215     //*============================================================================================
216     //*============================================================================================
217     //*============  END USER APPLICATION ========= END USER APPLICATION ====================
218     //*============================================================================================
219     //*============================================================================================
220 }
221
```