

```

1  //////////////////////////////////////
2  //CODI PROGRAMA WASPMOTE PORTA////////////////////////////////////
3  //Created By Rubén Almansa for EXIT Research group////////////////////////////////
4  //////////////////////////////////////
5  #include <WaspSensorEvent_v20.h>
6  #include <WaspXBeeZB.h>
7  #include <WaspFrame.h>
8  #include <WaspFrameConstants.h>
9
10 // Adreça Meshlium////////////////////////////////
11 char RX_ADDRESS[] = "0013A200408134BA";
12 //////////////////////////////////
13
14 //DEFINICIÓ VARIABLES COMUNS////////////////////////////////
15 uint8_t error,error2;
16 char* macHigh=""          ";
17 char* macLow=""          ";
18 timestamp_t time; // guarda el timestsmp del Meshlium
19 unsigned long epoch; // fa un break del timestamp i ho guarda en aquesta variable
20 int Tx; // resgistre el flag del frame enviat.
21 int noSend=0; //comptador d'enviaments erronis
22 int errorRTC=0; //registre el flag de la RTC ok.
23 int SumErrorRTC=0; // comptador d'errors de la sincronització deRTC
24 //////////////////////////////////
25
26 //VARIABLES PORTA////////////////////////////////
27 float temp;
28 int bat;
29 int P; // contacte de porta
30 float LDR;
31 float LDR2;
32 float V;
33 int PIR;
34 float Vs;// vibració
35 int Hum;
36
37 //////////////////////////////////
38 //*PROGRAMA D'INICI*////////////////////////////////
39 void setup()
40 {
41  /*ACTIVA la RTC i el port USB*/
42
43   RTC.ON();
44   delay(2000);
45   USB.ON();
46   delay(200);
47  /*S'activa la placa de sensor d'esdeveniments*/
48  SensorEventv20.ON();
49  delay(3000);
50  //Activa les interrupcions externes
51  SensorEventv20.setThreshold(SENS_SOCKET7,1);
52  SensorEventv20.setThreshold(SENS_SOCKET4,0.6);
53  enableInterrupts(SENS_INT); //habilitar les interrupcions de les sensorsboards
54
55   IniXBee();
56   checkNetworkParams();
57   ResetNetwork();
58   RTCMeshlium();
59  // Alarma que s'ativa quan esta a punt d'iniciar l'enviament
60  RTC.setAlarm1("00:00:08:00",RTC_OFFSET,RTC_ALM1_MODE4); //
61  delay(1000);
62 }
63
64 //////////////////////////////////
65 //*PROGRAMA PRINCIPAL*////////////////////////////////
66
67 void loop()
68 {
69  // interrupció externa PIR
70  if ((SensorEventv20.intFlag & SENS_SOCKET7))
71
72
73  {

```

```

74     intFlag &= ~(SENS_INT);
75     intFlag &= ~(SENS_SOCKET7);
76     USB.println("detecto");
77     PIR=PIR+1;
78 }
79 // interrupció extena vibracions
80 if (SensorEventv20.intFlag & SENS_SOCKET4)
81
82
83 {
84     intFlag &= ~(SENS_INT);
85     V=(SensorEventv20.readValue(SENS_SOCKET4)*1000); // possem per mil perquè sabem
      que son milivolts
86     Vs=1;// si hi ha alguna vibració molt elevada la registra
87 }
88
89
90 if(intFlag & RTC_INT )
91 {
92     disableInterrupts(SENS_INT);
93     intFlag &= ~(RTC_INT);
94     RTC.clearAlarmFlag();
95     delay(1000);
96     LlegirDades();
97     while (Tx==0){
98         epoch=RTC.getEpochTime();
99         RTC.breakTimeAbsolute( epoch, &time);
100         USB.println(RTC.getTime());
101         if(((time.minute == 00) || (time.minute == 10) || (time.minute == 20) || (time.minute
          == 30) || (time.minute == 40) || (time.minute == 50))&&Tx==0){EnviarDades();}
102         delay(5000);
103     }
104     Tx=0;
105     Vs=0;
106     PIR=0;
107     USB.println("enviat");
108     PWR.reboot();
109 }
110 // activa les interrupcions externes i dorm mentre no desperti la RTC o
      interrupcions externes
111 enableInterrupts(SENS_INT);
112 SensorEventv20.attachInt();
113 USB.println("DORMO");
114 PWR.sleep(SOCKET0_OFF);
115 RTC.ON();
116 delay(200);
117 SensorEventv20.detachInt();
118 SensorEventv20.loadInt();
119 }
120
121 //////////////////////////////////////
122 /*FUNCIÓ QUE LLEGEIX LES DADES */
123
124 void LlegirDades(){
125
126
127     temp = SensorEventv20.readValue(SENS_SOCKET5, SENS_TEMPERATURE);
128     delay(100);
129     bat=PWR.getBatteryLevel();
130     Hum = SensorEventv20.readValue(SENS_SOCKET6, SENS_HUMIDITY);
131     delay(1000);
132     LDR=SensorEventv20.readValue(SENS_SOCKET1);
133     delay(500);
134     // algoritme que llegeix la LDR
135     LDR2=53.125*log(LDR)+35,679;
136     USB.println(LDR2);
137     // dona sempre unvalor mínim de 1%
138     if(LDR2<0.5){LDR2=1;}
139     USB.println(LDR2);
140     P= SensorEventv20.readValue(SENS_SOCKET8);
141     if (Vs==0){
142         V=(SensorEventv20.readValue(SENS_SOCKET4)*1000);
143         delay(100);

```

```

144
145 // contacte magnetic de la porta, llegeix el voltatge del contacte i retorna un 1 o
146 // 0 depenen l'estat
147 if (P>1)
148 {
149     P=1;
150     delay(100);
151 }
152 else {P=0;}
153
154
155 ///////////////////////////////////////////////////
156 //*****AQUESTA FUNCIO S'ENCARREGA D'ENVIAR LES DADES*//////////
157
158 void EnviarDades(){
159
160     while(Tx==0){
161
162         frame.setID( macLow ); // és el identificador del WASPMOTE
163         frame.createFrame(ASCII); // indicador per iniciar el frame.
164         //s'afegeix les variables de cada waspmote que s'enviaràn
165         frame.addSensor(SENSOR_STR, temp,LDR2,V,P,PIR,Hum,bat);
166         delay(100);
167         // envia el paquet
168         error = xbeeZB.send( RX_ADDRESS, frame.buffer, frame.length );
169         if( error == 0 ) // comprova l'enviament
170
171         {
172             USB.println(F("send ok"));
173
174             // blink green LED
175             Utils.blinkGreenLED();
176             noSend=0;
177             Tx=1; // Variable per confirmar qu l'envio ha estat correcte
178             delay(60000);
179         }
180     else
181     {
182         USB.println(F("send error"));
183
184         if (noSend>10){PWR.reboot();}
185         noSend=noSend+1;
186     }
187 }
188
189
190 ///////////////////////////////////////////////////
191 //*****FUNCIO QUE CONNECTA AMB MESHLIUM PER OBTENIR EL TIMESTAMP*//////////
192
193 void RTCMeshlium()
194
195 {
196     while (errorRTC ==0){
197         error2 = xbeeZB.setRTCfromMeshlium(RX_ADDRESS);
198         // check flag si la RTC s'ha sincronitzat
199         if( error2 == 0 )
200         {
201             USB.print(F("SET RTC ok. "));
202             errorRTC=1;
203             USB.println(RTC.getTime());
204         }
205     else
206     {
207         USB.print(F("SET RTC error. "));
208         errorRTC=0;
209         SumErrorRTC=SumErrorRTC+1;
210         if (SumErrorRTC>10){PWR.reboot();} // si es produeixen molts errors reseteja el
211         sistema.
212     }
213 }
214

```

```

215 ///////////////////////////////////////////////////
216 //*****FUNCIÓ ENCARREGADA D'INICIAR ELS XBEE I OBTENIR LA MAC*/////////
217
218 void IniXBee ()
219
220 {
221     xbeeZB.ON();
222     delay(3000);
223     xbeeZB.getOwnMacLow(); // obté la MAC LOW
224     xbeeZB.getOwnMacHigh(); // obté la MAC HIGH
225     // Guarda les MAC
226     Utils.hex2str(xbeeZB.sourceMacHigh,macHigh,4);
227     Utils.hex2str(xbeeZB.sourceMacLow,macLow,4);
228 }
229
230 ///////////////////////////////////////////////////
231 //*****COMPROVA ELS PARAMETRES DE LA XARXA XBEE*/////////
232 void checkNetworkParams ()
233 {
234     // Obté la identificació de xarxa
235     xbeeZB.getOperating64PAN();
236
237     // comprova la associació amb meshlium
238     xbeeZB.getAssociationIndication();
239
240     while( xbeeZB.associationIndication != 0 )
241     {
242         delay(2000);
243
244
245         xbeeZB.getOperating64PAN();
246
247         USB.print(F("operating 64-b PAN ID: "));
248         USB.printHex(xbeeZB.operating64PAN[0]);
249         USB.printHex(xbeeZB.operating64PAN[1]);
250         USB.printHex(xbeeZB.operating64PAN[2]);
251         USB.printHex(xbeeZB.operating64PAN[3]);
252         USB.printHex(xbeeZB.operating64PAN[4]);
253         USB.printHex(xbeeZB.operating64PAN[5]);
254         USB.printHex(xbeeZB.operating64PAN[6]);
255         USB.printHex(xbeeZB.operating64PAN[7]);
256         USB.println();
257
258         xbeeZB.getAssociationIndication();
259     }
260
261     USB.println(F("\nJoined a network!"));
262
263
264     xbeeZB.getOperating16PAN();
265     xbeeZB.getOperating64PAN();
266     xbeeZB.getChannel();
267
268     USB.print(F("operating 16-b PAN ID: "));
269     USB.printHex(xbeeZB.operating16PAN[0]);
270     USB.printHex(xbeeZB.operating16PAN[1]);
271     USB.println();
272
273     USB.print(F("operating 64-b PAN ID: "));
274     USB.printHex(xbeeZB.operating64PAN[0]);
275     USB.printHex(xbeeZB.operating64PAN[1]);
276     USB.printHex(xbeeZB.operating64PAN[2]);
277     USB.printHex(xbeeZB.operating64PAN[3]);
278     USB.printHex(xbeeZB.operating64PAN[4]);
279     USB.printHex(xbeeZB.operating64PAN[5]);
280     USB.printHex(xbeeZB.operating64PAN[6]);
281     USB.printHex(xbeeZB.operating64PAN[7]);
282     USB.println();
283
284     USB.print(F("channel: "));
285     USB.printHex(xbeeZB.channel);
286     USB.println();
287

```

```
288 }
289 ///////////////////////////////////////////////////
290 //*****RESETEJA ELS PAREMETRES DE LA XARXA*/////////
291
292 void ResetNetwork(){
293
294     while((xbeeZB.channel) !=0x10){
295         xbeeZB.OFF();
296         delay(1000);
297         xbeeZB.ON();
298         delay(3000);
299         xbeeZB.resetNetwork(0); // reseteja els paràmetres de la xarxa
300         delay(1000);
301         checkNetworkParams(); // comprova aquests paràmetres.
302     }
303 }
304
305
```