

```

1  //////////////////////////////////////
2  //CODI PROGRAMA WASPMOTE SmartCities////////////////////////////////////
3  //Created By Rubén Almansa for EXIT Research group////////////////////////////////
4  //////////////////////////////////////
5
6  #include <WaspSensorCities.h>
7  #include <WaspXBeeZB.h>
8  #include <WaspFrame.h>
9  #include <WaspFrameConstants.h>
10
11  //Adreça Meshlium////////////////////////////////
12  char RX_ADDRESS[] = "0013A200408134BA";
13  //////////////////////////////////
14
15  //DEFINICIÓ VARIABLES COMUNS////////////////////////////////
16  uint8_t error,error2;
17  char* macHigh="";
18  char* macLow="";
19  timestamp_t time; // guarda el timestsmp del Meshlium
20  unsigned long epoch; // fa un break del timestamp i ho guarda en aquesta variable
21  int Tx; // registre el flag del frame enviat.
22  int noSend=0; //comptador d'enviaments erronis
23  int errorRTC=0; //registre el flag de la RTC ok.
24  int SumErrorRTC=0; // comptador d'errors de la sincronització deRTC
25  //////////////////////////////////
26
27  //VARIABLES SMART CITIES////////////////////////////////
28  float temp;
29  int bat; // nivell de bateria
30  double SorollMax,SorollMaxAct; // variables
31  float Soroll; // Soroll en l'instant d'adquirir
32  float Leq=0; // resultat final del soroll Mig
33  float Ldr;
34  double SorollAvg=0, SorollAvg1=0 ;// variables per els càlculs soroll són per fer la
    mitjana
35  int ad=0; // index de la taula de registres del valors de so
36  int b=0, c=0, ini=1,d=0;
37  double S[500]; // taula de registres del so
38  double logS;
39  float loglux,Lux; // càlcul del luxs
40  //////////////////////////////////
41  //PROGRAMA D'INICI////////////////////////////////
42  void setup()
43  {
44  /*ACTIVA la RTC i el port USB*/
45
46      RTC.ON();
47      delay(2000);
48      USB.ON();
49      delay(200);
50
51      SensorCities.ON(); // Activa la sensorboard
52      delay(2000);
53
54      IniXBee();
55      checkNetworkParams();
56      ResetNetwork();
57      RTCMeshlium();
58
59  }
60
61  //////////////////////////////////
62  //PROGRAMA PRINCIPAL////////////////////////////////
63
64  void loop()
65  {
66
67      USB.println(RTC.getTime());
68      LlegirSO();
69      epoch=RTC.getEpochTime(); // guarda el time en un format, és necessari per la
        comanda següent
70      RTC.breakTimeAbsolute( epoch, &time); // trenca el timestamp per obtenir el minut
        que estem

```

```

71
72 //Aquest "if" s'encarrega de comprovar el minut d'enviament
73 if(((time.minute == 00) || (time.minute == 10)|| (time.minute == 20)|| (time.minute
== 30) || (time.minute == 40)|| (time.minute == 50))&&Tx==0){
74
75     LlegirDades();
76     calculdBA();
77     EnviarDades();
78     ad=0;
79     SorollAvg=0;
80     PWR.reboot();
81 }
82 }
83
84
85 ///////////////FUNCIÓ QUE LLEGEIX EL SO////////////////////
86 void LlegirSO(){
87
88     // encén el sensor d'audio
89     SensorCities.setSensorMode(SENS_ON, SENS_CITIES_AUDIO);
90     delay(2000);
91
92     // llegeix sensor d'audio
93     Soroll = SensorCities.readValue(SENS_CITIES_AUDIO);
94
95     // apaga el sensor d'audio
96     SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_AUDIO);
97
98     USB.print(F("Sound pressure: "));
99     USB.print(Soroll);
100     USB.println(F("dBA"));
101     S[ad]=Soroll;           // guarda els valors adquirits en una taula
102     USB.println(S[ad]);
103     ad=ad+1;               // incrementa l'index del valor d'audio
104
105     delay(1000);
106 }
107
108 ///////////////////////////////////////////////////
109 /*FUNCIÓ QUE LLEGEIX LES DADES RESTANTS AL SO */
110
111 void LlegirDades()
112 {
113
114     SensorCities.ON();
115     delay(2000);
116     USB.println("llegir");
117     bat=PWR.getBatteryLevel();
118     SensorCities.setSensorMode(SENS_ON, SENS_CITIES_TEMPERATURE);
119     delay(1500);
120     temp = SensorCities.readValue(SENS_CITIES_TEMPERATURE);
121     SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_TEMPERATURE);
122     SensorCities.setSensorMode(SENS_ON, SENS_CITIES_LDR);
123     delay(1500);
124     Ldr= SensorCities.readValue(SENS_CITIES_LDR);
125     SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_LDR);
126     SensorCities.setSensorMode(SENS_ON, SENS_CITIES_HUMIDITY);
127     delay(5000);
128     Lux=analogRead(ANALOG7);
129     SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_HUMIDITY);
130     delay(100);
131     Lux=(Lux*5/1024);
132     loglux=pow(10, Lux); // càlcul logarítmic dels luxs
133     USB.println(loglux);
134
135 }
136 ///////////////////////////////////////////////////
137 /** FUNCIÓ QUE FA EL CALCUL MIG DEL SO I BUSCA EL VALOR MÀXIM */
138
139 void calculdBA(){
140
141     d=ad; //
142     b=ad-1; //

```

```

143     ad =0; //
144     USB.println(b);
145     SorollMax=S[ad];
146
147     /*llegim tots els registres que hem guardat a S[ad], fem la suma
148     i comprovem quin és el màxim */
149     for (int e=0; e<=b;e++){
150
151         if(SorollMax<S[ad]){SorollMax=S[ad];}
152         SorollAvg1=pow(10,(S[ad]/10));
153         USB.println (SorollAvg1);
154         SorollAvg=SorollAvg1+SorollAvg;
155         USB.println (SorollAvg);
156         ad=ad+1;
157     }
158
159     USB.println(d);
160     logS=(SorollAvg/d); // Promig del SO per la formula
161     USB.println(logS);
162     Leq=10*log10(logS); // Resultat final del SO
163     USB.println(Leq);
164     if(Leq<50){Leq=50;SorollMax=50;}
165
166 }
167
168 //////////////////////////////////////
169 //*****AQUESTA FUNCIO S'ENCARREGA D'ENVIAR LES DADES*////////
170
171 void EnviarDades(){
172
173     while(Tx==0){
174
175         frame.setID( macLow ); // és el identificador del WASPMOTE
176         frame.createFrame(ASCII); // indicador per iniciar el frame.
177         //s'afegeix les variables de cada waspmote que s'enviaràn
178         frame.addSensor(SENSOR_STR, temp,Ldr,Leq,SorollMax,loglux,bat);
179         delay(100);
180         // envia el paquet
181         error = xbeeZB.send( RX_ADDRESS, frame.buffer, frame.length );
182         if( error == 0 ) // comprova l'enviament
183
184         {
185             USB.println(F("send ok"));
186
187             // blink green LED
188             Utils.blinkGreenLED();
189             noSend=0;
190             Tx=1; // Variable per confirmar qu l'envio ha estat correcte
191             delay(60000);
192         }
193         else
194         {
195             USB.println(F("send error"));
196
197             if (noSend>10){PWR.reboot();}
198             noSend=noSend+1;
199         }
200     }
201 }
202
203 //////////////////////////////////////
204
205 //***** FUNCIO QUE CONNECTA AMB MESHLIUM PER OBTENIR EL TIMESTAMP*////////
206
207 void RTCMeshlium()
208
209 {
210     while (errorRTC ==0){
211         error2 = xbeeZB.setRTCfromMeshlium(RX_ADDRESS);
212         // check flag si la RTC s'ha sincronitzat
213         if( error2 == 0 )
214         {
215             USB.print(F("SET RTC ok. "));

```

```

216     errorRTC=1;
217     USB.println(RTC.getTime());
218 }
219 else
220 {
221     USB.print(F("SET RTC error. "));
222     errorRTC=0;
223     SumErrorRTC=SumErrorRTC+1;
224     if (SumErrorRTC>10){PWR.reboot();} // si es produeixen molts errors resetja el
        sistema.
225 }
226 }
227 }
228 ///////////////////////////////////////////////////
229 //*****FUNCIÓ ENCARREGADA D'INICIAR ELS XBEE I OBTENIR LA MAC*/////////
230
231 void IniXBee()
232
233 {
234     xbeeZB.ON();
235     delay(3000);
236     xbeeZB.getOwnMacLow(); // obté la MAC LOW
237     xbeeZB.getOwnMacHigh(); // obté la MAC HIGH
238     // Guarda les MAC
239     Utils.hex2str(xbeeZB.sourceMacHigh,macHigh,4);
240     Utils.hex2str(xbeeZB.sourceMacLow,macLow,4);
241 }
242
243 ///////////////////////////////////////////////////
244 //*****COMPROVA ELS PARAMETRES DE LA XARXA XBEE*/////////
245 void checkNetworkParams()
246 {
247     // Obté la identificació de xarxa
248     xbeeZB.getOperating64PAN();
249
250     // comprova la associació amb meshlium
251     xbeeZB.getAssociationIndication();
252
253     while( xbeeZB.associationIndication != 0 )
254     {
255         delay(2000);
256
257
258         xbeeZB.getOperating64PAN();
259
260         USB.print(F("operating 64-b PAN ID: "));
261         USB.printHex(xbeeZB.operating64PAN[0]);
262         USB.printHex(xbeeZB.operating64PAN[1]);
263         USB.printHex(xbeeZB.operating64PAN[2]);
264         USB.printHex(xbeeZB.operating64PAN[3]);
265         USB.printHex(xbeeZB.operating64PAN[4]);
266         USB.printHex(xbeeZB.operating64PAN[5]);
267         USB.printHex(xbeeZB.operating64PAN[6]);
268         USB.printHex(xbeeZB.operating64PAN[7]);
269         USB.println();
270
271         xbeeZB.getAssociationIndication();
272     }
273
274     USB.println(F("\nJoined a network!"));
275
276
277     xbeeZB.getOperating16PAN();
278     xbeeZB.getOperating64PAN();
279     xbeeZB.getChannel();
280
281     USB.print(F("operating 16-b PAN ID: "));
282     USB.printHex(xbeeZB.operating16PAN[0]);
283     USB.printHex(xbeeZB.operating16PAN[1]);
284     USB.println();
285
286     USB.print(F("operating 64-b PAN ID: "));
287     USB.printHex(xbeeZB.operating64PAN[0]);

```

```
288     USB.println(xbeeZB.operating64PAN[1]);
289     USB.println(xbeeZB.operating64PAN[2]);
290     USB.println(xbeeZB.operating64PAN[3]);
291     USB.println(xbeeZB.operating64PAN[4]);
292     USB.println(xbeeZB.operating64PAN[5]);
293     USB.println(xbeeZB.operating64PAN[6]);
294     USB.println(xbeeZB.operating64PAN[7]);
295     USB.println();
296
297     USB.print(F("channel: "));
298     USB.println(xbeeZB.channel);
299     USB.println();
300
301 }
302 //////////////////////////////////////////////////
303 //*****RESETEJA ELS PAREMETRES DE LA XARXA*////////
304
305 void ResetNetwork(){
306
307     while((xbeeZB.channel) !=0x10){
308         xbeeZB.OFF();
309         delay(1000);
310         xbeeZB.ON();
311         delay(3000);
312         xbeeZB.resetNetwork(0); // reseteja els paràmetres de la xarxa
313         delay(1000);
314         checkNetworkParams(); // comprova aquests paràmetres.
315     }
316 }
317
318
```