

```

1  //////////////////////////////////////
2  //CODI PROGRAMA WASPMOTE GASOS////////////////////////////////////
3  //Created By Rubén Almansa for EXIT Research group////////////////////////////////
4  //////////////////////////////////////
5
6  #include <WaspSensorGas_v20.h>
7  #include <WaspXBeeZB.h>
8  #include <WaspFrame.h>
9  #include <WaspFrameConstants.h>
10 #define GAIN 7 //GAIN del sensor CO2
11 #define cal 3.13 // aprox 350-400ppm CO2
12
13 // Adreça Meshlium////////////////////////////////
14 char RX_ADDRESS[] = "0013A200408134BA";
15 //////////////////////////////////////
16
17 //DEFINICIÓ VARIABLES COMUNS////////////////////////////////
18 uint8_t error,error2;
19 char* macHigh=" ";
20 char* macLow=" ";
21 timestamp_t time; // guarda el timestsmp del Meshlium
22 unsigned long epoch; // fa un break del timestamp i ho guarda en aquesta variable
23 int Tx; // registre el flag del frame enviat.
24 int noSend=0; //comptador d'enviaments erronis
25 int errorRTC=0; //registre el flag de la RTC ok.
26 int SumErrorRTC=0; // comptador d'errors de la sincronització deRTC
27 //////////////////////////////////////
28
29 //VARIABLES GASOS////////////////////////////////
30 float temp;
31 int bat;
32 float CO2=0,CO20=0; // valors intermitjos de càlcul
33 float pressio;
34 float ppm; // valor final de CO2 en ppm
35 float CO22=0; // valor final de CO2 en volts
36 float tempExt; // sonda exterior
37 float buit=0;
38 int coDiv=0; // numero de dades CO2 adquirides
39 float oPppm=0;
40 int tmp =0;
41 //////////////////////////////////////
42 //PROGRAMA D'INICI////////////////////////////////
43 void setup()
44 {
45 /*ACTIVA la RTC i el port USB*/
46
47 RTC.ON();
48 delay(2000);
49 USB.ON();
50 delay(200);
51
52 SensorGasv20.ON(); // Activa la sensor board
53 SensorGasv20.configureSensor(SENS_CO2, GAIN); // es configura el guany d'adquisició
54
55
56 IniXBee();
57 checkNetworkParams();
58 ResetNetwork();
59 RTCMeshlium();
60
61 }
62
63 //////////////////////////////////////
64 //PROGRAMA PRINCIPAL////////////////////////////////
65
66 void loop()
67 {
68
69 USB.print("agafo mesures");
70 LlegirDades();
71 LlegirCO2();
72
73 /* despres de llegir totes les dades espera que arribi el temps per enviar les dades*/

```

```

74 while (Tx==0){
75   epoch=RTC.getEpochTime();
76   RTC.breakTimeAbsolute( epoch, &time);
77   USB.println(RTC.getTime());
78   if(((time.minute == 00) || (time.minute == 10) || (time.minute == 20) || (time.minute ==
30) || (time.minute == 40) || (time.minute == 50))&&Tx==0){EnviarDades();}
79   delay(5000);
80
81 }
82 /*un cop envia les dades fa un repòs de 3 minuts i torna a començar el inici*/
83 USB.println("a dormir");
84 PWR.deepSleep("00:00:03:00",RTC_OFFSET,RTC_ALM1_MODE4,SOCKET0);
85 RTC.ON();
86 delay(100);
87 intFlag &= ~(RTC_INT);
88 RTC.clearAlarmFlag();
89 delay(1000);
90 PWR.reboot();
91 }
92
93
94 ///////////////FUNCIÓ QUE LLEGEIX EL CO2////////////////////
95
96 void LLegirCO2 (){
97
98   USB.println("Activa CO2");
99   SensorGasv20.ON();
100  SensorGasv20.configureSensor(SENS_CO2, 7);
101  SensorGasv20.setSensorMode(SENS_ON, SENS_CO2);
102  delay(40000);
103
104
105  for(int co=0;co<420;co++){ // fa un promig promig de 3.5 minuts de dades adquirides
106    CO20= SensorGasv20.readValue(SENS_CO2);
107    USB.println(CO20);
108    if(CO20<3.13){
109      CO2 = CO2 + CO20;
110      coDiv=coDiv+1;}
111    delay(500);
112  }
113  USB.println(coDiv);
114
115  SensorGasv20.setSensorMode(SENS_OFF, SENS_CO2);
116  delay(1000);
117  SensorGasv20.OFF();
118  delay(100);
119  USB.println(tempExt);
120  USB.print("acabo mesures");
121  CO22=CO2/coDiv;
122  // marca el slimits d'error
123  if(CO22<3.00 || CO22 >3.15 || coDiv ==0){CO22=3.15;}
124  USB.println(CO22);
125  oPppm=(((cal-CO22)*1000)+158.631)/62.877); // equacio logaritmica dels ppm
126  ppm=pow(10,oPppm);
127  /*Marca els limits superior e inferior*/
128  if (ppm>10000){ppm=10000;}
129  if (ppm<350) {ppm=350;}
130  }
131
132  //////////////////////
133  /*FUNCIÓ QUE LLEGEIX LES DADES RESTANTS AL CO2 */
134
135  void LlegirDades (){
136
137    SensorGasv20.ON();
138    delay(3000);
139    SensorGasv20.setSensorMode(SENS_ON,SENS_TEMPERATURE);
140    delay(2000);
141    temp = SensorGasv20.readValue(SENS_TEMPERATURE);
142    SensorGasv20.setSensorMode(SENS_OFF,SENS_TEMPERATURE);
143    delay(2000);
144    SensorGasv20.setSensorMode(SENS_ON, SENS_PRESSURE);
145    delay(1000);

```

```

146 pressio=SensorGasv20.readValue(SENS_PRESSURE);
147 delay(500);
148 SensorGasv20.setSensorMode(SENS_OFF, SENS_PRESSURE);
149 bat=PWR.getBatteryLevel();
150 tempExt = Utils.readTempDS1820(DIGITAL1,TRUE); // comanda per one-wire
151 if(tempExt!=-1000){Utils.writeEEPROM(1025,tempExt);}
152 /* com a vegades una sola lectura dona errors, en cas d'error mes de 10 errors
donarà com a respota el valor anterior*/
153 while((tempExt== -1000) && (tmp<10)){ tempExt =
Utils.readTempDS1820(DIGITAL1,TRUE);tmp=tmp+1;delay(200);
154 USB.println(tempExt);}
155 if(tempExt== -1000){tempExt=Utils.readEEPROM(1025);}
156 USB.println(tempExt);
157 }
158
159 ///////////////////////////////////////////////////
160 //*****AQUESTA FUNCIÓ S'ENCARREGA D'ENVIAR LES DADES*//////////
161
162 void EnviarDades(){
163
164     while(Tx==0){
165
166         frame.setID( macLow ); // és el identificador del WASPMOTE
167         frame.createFrame(ASCII); // indicador per iniciar el frame.
168         //s'afegeix les variables de cada waspmote que s'enviaràn
169         frame.addSensor(SENSOR_STR, temp,pressio,tempExt,CO22,ppm,bat);
170         delay(100);
171         // envia el paquet
172         error = xbeeZB.send( RX_ADDRESS, frame.buffer, frame.length );
173         if( error == 0 ) // comprova l'enviament
174
175         {
176             USB.println(F("send ok"));
177
178             // blink green LED
179             Utils.blinkGreenLED();
180             noSend=0;
181             Tx=1; // Variable per confirmar qu l'envio ha estat correcte
182             delay(60000);
183         }
184         else
185         {
186             USB.println(F("send error"));
187
188             if (noSend>10){PWR.reboot();}
189             noSend=noSend+1;
190         }
191     }
192 }
193
194 ///////////////////////////////////////////////////
195
196 //***** FUNCIÓ QUE CONNECTA AMB MESHLIUM PER OBTENIR EL TIMESTAMP*//////////
197
198 void RTCMeshlium()
199
200 {
201     while (errorRTC ==0){
202         error2 = xbeeZB.setRTCfromMeshlium(RX_ADDRESS);
203         // check flag si la RTC s'ha sincronitzat
204         if( error2 == 0 )
205         {
206             USB.print(F("SET RTC ok. "));
207             errorRTC=1;
208             USB.println(RTC.getTime());
209         }
210         else
211         {
212             USB.print(F("SET RTC error. "));
213             errorRTC=0;
214             SumErrorRTC=SumErrorRTC+1;
215             if (SumErrorRTC>10){PWR.reboot();} // si es produeixen molts errors reseteja el
sistema.

```

```

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

```

```
289     USB.printHex(xbeeZB.channel);
290     USB.println();
291
292 }
293 ///////////////////////////////////////////////////////////////////
294 //*****RESETEJA ELS PAREMETRES DE LA XARXA*/////////////////
295
296 void ResetNetwork(){
297
298     while((xbeeZB.channel) !=0x10){
299         xbeeZB.OFF();
300         delay(1000);
301         xbeeZB.ON();
302         delay(3000);
303         xbeeZB.resetNetwork(0); // reseteja els paràmetres de la xarxa
304         delay(1000);
305         checkNetworkParams(); // comprova aquests paràmetres.
306     }
307 }
308
309
```