empierre / arduino ★ Star 5 Fork 0

branch: master arduino / AirQuality-MQ135.ino


empierre on 3 Dec 2014 Rename MQ135_2.ino to AirQuality-MQ135.ino

1 contributor

115 lines (92 sloc) 3.446 kb Raw Blame History

```
/*
  Arduino MQ135

  connect the sensor as follows :

  A H A   >>> 5V
  B       >>> A0
  H       >>> GND
  B       >>> 10K ohm >>> GND

  Contribution: epierre
  Based on David Gironi http://davidegironi.blogspot.fr/2014/01/cheap-co2-meter-using-m
  http://skylink.dl.sourceforge.net/project/davidegironi/avr-lib/avr_lib_mq135_01.zip

*/

#include <SPI.h>
#include <MySensor.h>
#include <Wire.h>

#define CHILD_ID_AIQ 0
#define AIQ_SENSOR_ANALOG_PIN 0

#define MQ135_DEFAULTPPM 399 //default ppm of CO2 for calibration
#define MQ135_DEFAULTRO 68550 //default Ro for MQ135_DEFAULTPPM ppm of CO2
#define MQ135_SCALINGFACTOR 116.6020682 //CO2 gas value

#define MQ135_EXPONENT -2.769034857 //CO2 gas value
#define MQ135_MAXRSRO 2.428 //for CO2
#define MQ135_MINRSRO 0.358 //for CO2

unsigned long SLEEP_TIME = 30000; // Sleep time between reads (in seconds)
//VARIABLES
//float Ro = 10000.0;    // this has to be tuned 10K Ohm
float mq135_ro = 10000.0;    // this has to be tuned 10K Ohm
int val = 0;           // variable to store the value coming from the sensor
float valAIQ =0.0;
float lastAIQ =0.0;

MySensor gw;
MyMessage msg(CHILD_ID_AIQ, V_VAR1);

void setup()
{
  gw.begin();

```

```
  // Send the sketch version information to the gateway and Controller
  gw.sendSketchInfo("AIQ Sensor MQ135", "1.0");

  // Register all sensors to gateway (they will be created as child devices)
  gw.present(CHILD_ID_AIQ, S_AIR_QUALITY);

}

/*
 * get the calibrated ro based upon read resistance, and a know ppm
 */
long mq135_getro(long resvalue, double ppm) {
return (long)(resvalue * exp( log(MQ135_SCALINGFACTOR/ppm) / MQ135_EXPONENT ));
}

/*
 * get the ppm concentration
 */
double mq135_getppm(long resvalue, long ro) {
double ret = 0;

double validinterval = 0;
validinterval = resvalue/(double)ro;
if(validinterval<MQ135_MAXRSRO && validinterval>MQ135_MINRSRO) {
ret = (double)MQ135_SCALINGFACTOR * pow( ((double)resvalue/ro), MQ135_EXPONENT);
}
return ret;
}

void loop()
{
  uint16_t valr = analogRead(AIQ_SENSOR_ANALOG_PIN);// Get AIQ value
  Serial.println(val);
  uint16_t val =  ((float)22000*(1023-valr)/valr);
  mq135_ro = mq135_getro(val, MQ135_DEFAULTPPM);
  //convert to ppm (using default ro)
  valAIQ = mq135_getppm(val, MQ135_DEFAULTRO);

  Serial.print ( "Vrl / Rs / ratio:");
  Serial.print ( val);
  Serial.print ( " / ");
  Serial.print ( mq135_ro);
  Serial.print ( " / ");
  Serial.print ( valAIQ);


  if (valAIQ != lastAIQ) {
      gw.send(msg.set((int)ceil(valAIQ)));
      lastAIQ = ceil(valAIQ);
  }

  // Power down the radio.  Note that the radio will get powered back up
  // on the next write() call.
  gw.sleep(SLEEP_TIME); //sleep for: sleepTime
}


/****************************  MQGetPercentage ********************************
Input:   rs_ro_ratio - Rs divided by Ro
```

```
104           pcurve      - pointer to the curve of the target gas
105   Output:   ppm of the target gas
106   Remarks:  By using the slope and a point of the line. The x(logarithmic value of ppm)
107           of the line could be derived if y(rs_ro_ratio) is provided. As it is a
108           logarithmic coordinate, power of 10 is used to convert the result to non-logar
109           value.
110   ************************************************************************************/
111   int  MQGetPercentage(float rs_ro_ratio, float ro, float *pcurve)
112   {
113     return (double)(pcurve[0] * pow(((double)rs_ro_ratio/ro), pcurve[1]));
114   }
```