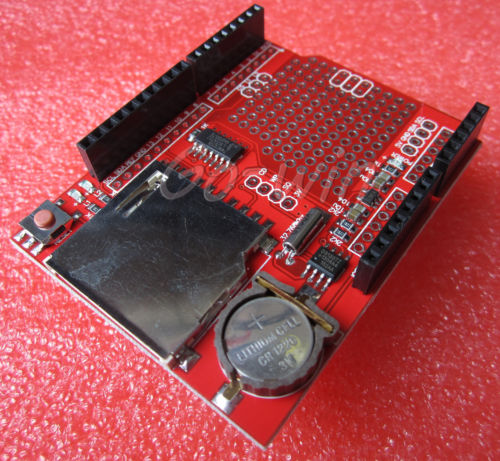
**20150429 CO2 Gas Sensor with Data Logging using the MQ-135 & Arduino Uno**

Let’s get the easy part of this project out of the way 1st.

Data Logging, I bought this shield off eBay for $6.37 - it has a small prototyping area, a DS1307 real time clock (Clock was set when I got it, and it had a battery in it) and a SD card reader/writter



|  |
| --- |

<http://www.ebay.com/itm/181586356162?_trksid=p2057872.m2749.l2649&ssPageName=STRK%3AMEBIDX%3AIT>

The SD card seemed to be limited to reading upto 2gb cards, not the SDHC cards, this is fine, 2Gb is really a lot of information for this.

I found this sketch that works with this

<https://github.com/tigoe/DataloggingExamples/blob/master/SDCardDataLoggerWithRTC/SDCardDataLoggerWithRTC.ino>

(They were doing temperature sensing with a BMP085, so easy to replace the code for the BMP and put the code for the MQ135) Other than taking out the BMP085 code, I had to change the CS PIN for the SD card reader. The SD card is on the SPI bus, on this shield it’s hooked up to the common SPI pins (MOSI 11, MISO 12, CLK 13, CS 10). The DS1307 RTC is hooked up to the I2C pins (SCL A4, SDA A5)

So, easy. The sketch is pretty well documented and it was pretty easy to figure out what needed to be changed, and what didn’t.

This sketch works as written (at least for a short time), but there are actually 2 errors, one was a pretty large logic error, the other was a minor formatting problem with the data file.

The formatting error, was pretty easy to correct - The author of the sketch has the data file formatted as a CSV, which is fine, and worked, it even imported the data into the LibreOffice spreadsheet just fine, but that is when I noticed that each time the sketch starts it writes column headers. I removed this from the sketch, and changed the file to just a standard text file. (It would be easy enough to change it back to a CSV file thou, without column headers)

I saw the logic error almost instantly, and the fix pretty simple. So I’ll show the parts of the code that have the error, and see if you see it.

const int interval = 10; // the interval between reads, in seconds

Inside void loop() { }

if (now.second() > lastReadTime + interval) {

…….

lastReadTime = now.second();

}

Do you see the problem?

So the way it is written it will work as long as seconds is less than 49.

So after a few reads of the sensor, it will stop, and needs to be reset.

This is a big logic error, you shouldn’t be counting the now.seconds, you should be counting how much time has passed.

The fixed code looks like this:

Inside void loop () {}

currentmillis = millis();

// if ten seconds have passed:

if ((currentmillis - lastReadTime) > (interval \* 1000)) {

lastReadTime = currentmillis;

…….

}

As you can see I’m looking at how much time has passed, don’t care about the seconds at all.

Did have to add an extra variable to do this, but it works. In my sketch I have it reading every 30 seconds.

One last problem with this sketch, and I haven’t found any way around it, is if you turn off the UNO while it’s writting to the SD card you’ll have a corrupt SD card, this happen to me twice, so be care!

---- And Now for the much more complicated part of this demo, the CO2 Sensor.

One of the things about this sensor is getting it calibrated, and I did find some information about doing it, but I don’t think I have mine calibrated right.

<http://davidegironi.blogspot.com/2014/01/cheap-co2-meter-using-mq135-sensor-with.html#.VT2HueS5Qrg>

The above blog posting explains the math pretty well, but it was above my head.

The MQ-135 looks like it can sense several different types of gas, and it all depends on the resistance of the heater, on a curve with the load resistance.

The data sheet can be found here:

<https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf>

Now this is the data sheet for just the sensor, I bought it on a board, and I couldn’t find the data sheet for it on the board



The blog post has a similar idea on how to use this as I did. As he points out the data sheet is not overly helpful. He seemed to think if you know what the PPM levels are you can adjust your sensor for that. And I am thinking the same, if you have a CO2 meter that gives a read out, you should beable to adjust your resistance values in your sketch and come close.

<http://co2now.org/Current-CO2/CO2-Now/>

Has the Atmospheric CO2 levels right around 400ppm. And I was able to get mine to read pretty consistently at 399ppm, is it right (??????) It could be completely wrong for all I know, and it may very well be wrong.

The writer of the blog does explain his reasoning a bit more with some math behind it. I am much simpler, and for the purposes of the demo, I didn’t go crazy trying to figure out 100% - I figure if it’s close, it’s good enough - I would love to beable to verify that I am getting good readings, but.

I did leave it on for 24 hours and let the heater get nice an hot, and after the 24 hour period it did seem to have some slightly better results.

So here are a few more resources I found on these sensors:

<http://playground.arduino.cc/Main/MQGasSensors>

<https://www.olimex.com/Products/Components/Sensors/_resources/SNS-MQ-sensors.pdf>

<http://www.futurlec.com/Datasheet/Sensor/MQ-135.pdf> (same data sheet from before)

General Information:

Carbon Dioxide CO2

<https://www.dhs.wisconsin.gov/chemical/carbondioxide.htm>

Safe Levels of CO2:

<https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms>

From this list it looks like anything above 1000ppm will cause problems, with anything about 5000 starting to get into the dangerous area. And 40,000 ppm and above, death!

The sketch I ended up finding that seemed to work was also based on the work of the blogger (David Gironi) and I found the sketch on codebender. Made a few changes to some of the values, and added it to the Data Logger Sketch, otherwise, it was pretty much unmodified.

My Sketch:

<https://codebender.cc/sketch:109629>