Shinyei Dust Sensor Air Quality Monitoring
With TI Launchpad MSP430G2

Objective: To know when I need to clean my room. I usually vacuum every two or three weeks, but sometimes am lazy and I need a reminder. The dust sensor will be more motivation to get me to clean, when I see the dust levels increase.

Table of Contents:

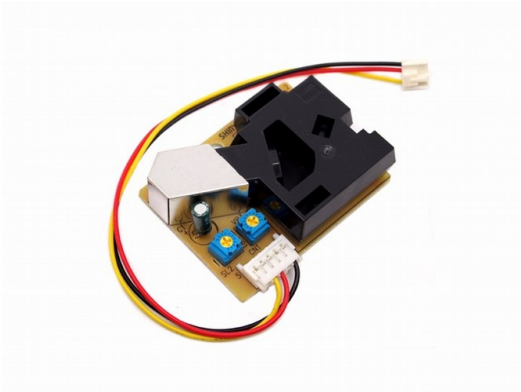


Figure 1: Shinyei PPD42NS Air Quality

Parts List:

- Shinyei PPD42NS Air Quality Sensor
- MSP-EXP430G2553 V1.5
- 5V Adapter (required as MSP is 3.3V only)
- SD Card adapter
- Ethernet Adapter
- (optional) 16x2 LCD
- Energia version 0101E0012 (or later, possibly)

Work Log

Here are some things I've learned as I've worked on this project.

IDE
I'm using Energia from energia.nu which is an Arduino IDE clone for TI Launchpads. This will allow for rapid development. And ease me into the TI platform.

MSP-EXP430G2 V1.5
There are different version of this. My particular board is the
and as explained here: http://energia.nu/pin-maps/guide_msp430g2launchpad/ and
http://energia.nu/pin-maps/guide_msp430g2launchpad/
You need to rotate the UART jumpers to get UART to display correctly. Oddly enough you can choose a different chip in the boards list and serial will print out right... But switch the jumpers and the correct board will work with UART. Otherwise, it does not work out of the box for serial.print. Though the blink example sketch works.

Dust Sensor
The code for the dust sensor is found online easily. The pinout is tricky as the colours of the wire are nonsense, but the pinout seems to be the same for all sensors, and is: PIN 1 (closest to black box) GND, PIN 3, VDD (5+V), PIN 4 output A. there is also an output B for different readings (I think size) of dust. I'm not that particular (no pun intended) about my dust so I will go with the one most people are using.

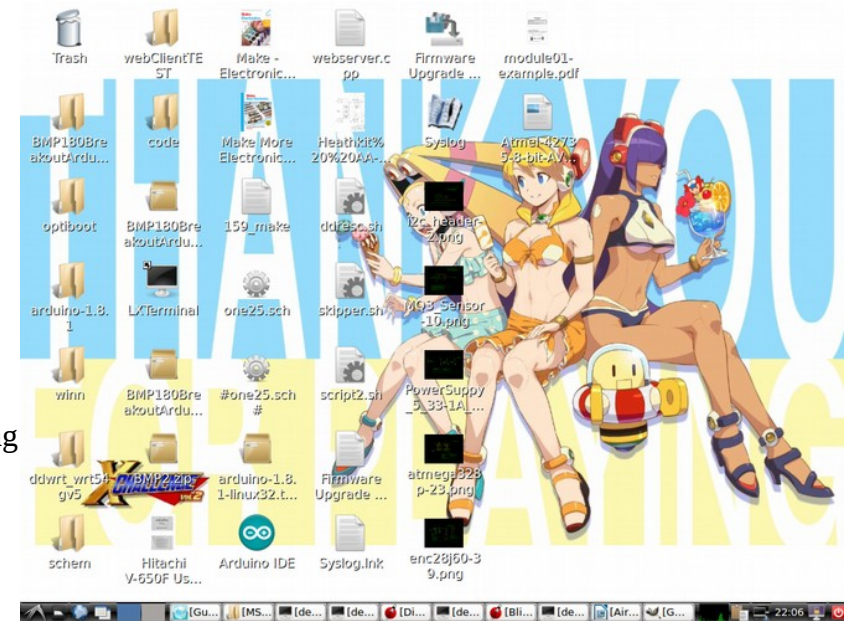
3.3V only on TI
The TI takes input of USB but only outputs 3.3V. Fail. I'll through a 5V PSU on the board. It won't be connected to USB for its use anyways.

POW function
There is a pow function (power exponent) in arduino. In TI, I changed it to powf, and included math.h.

I'm not sure if my change was correct.

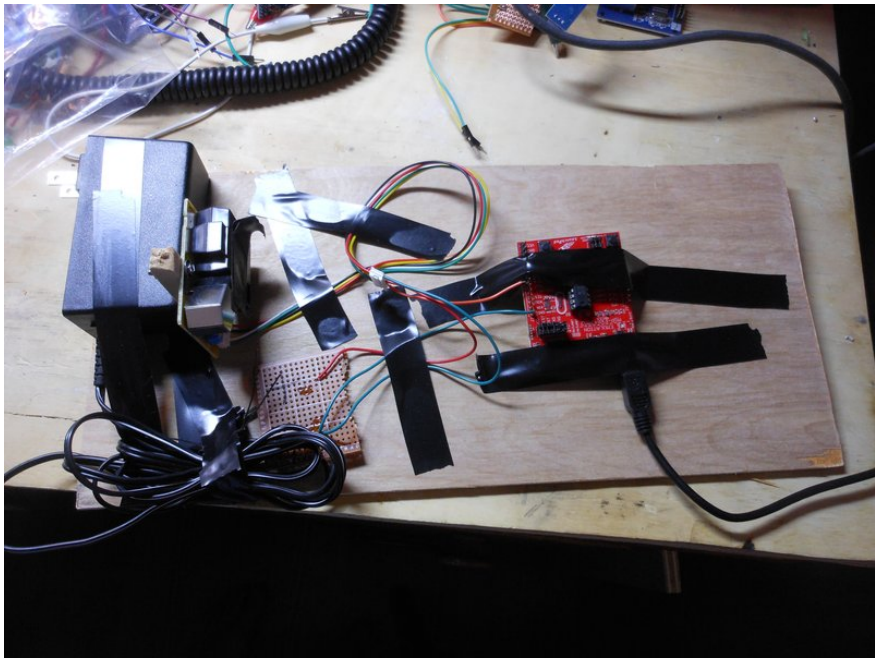
EDIT: looks like there is a LED tied to pin 14. I'm moving to pin 13. I meant to use 13 earlier but accidentally used the wrong pin and kept using it. Oops.
EDIT 2: Actually, I was reading from the wrong pin in software. Maybe I will leave it on the LED pin as you can see when the dust is firing off the sensor. Hm.... Neat.

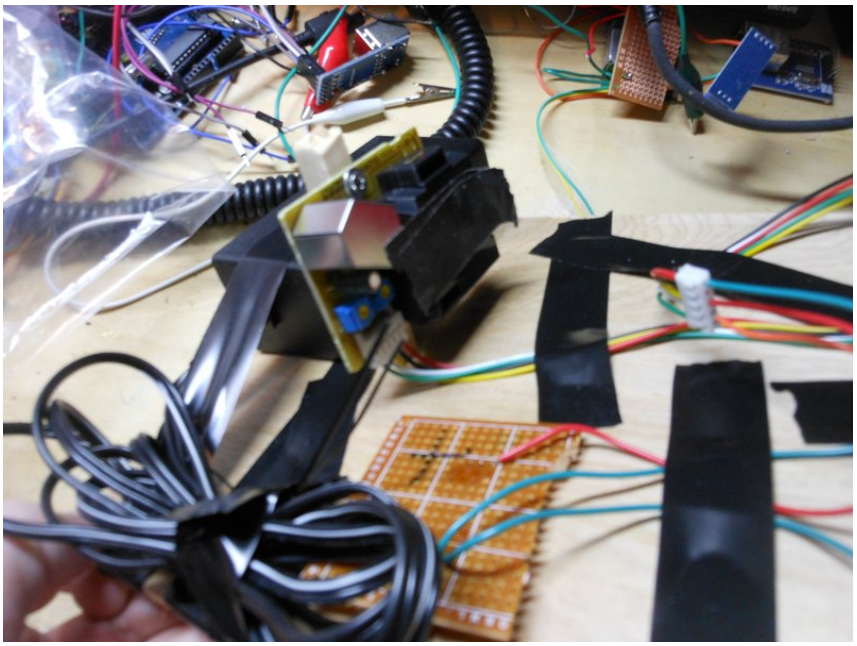
EDIT 3: Yes, so I had to use powf instead of pow, and it works. The accidental incorrect pin was a bonus as it allows me to see visually how often the sensor is going off. Over time, I should have a vague grasp of the dust levels just looking at the light.



This is an aside, but I realized today that happy pictures make me happy, so I should surround myself with pleasant things. I often keep a blank background, but I realized, that a cute picture will make me happier than a blank or default background. It's the little things. I digress.

Work Log 08/2018
I've put everything on a single piece of plywood, and need to do some more work on the code. I've decided to forgo the SD card, as I don't want to deal with reading a 2MB sd card with the buffer provided by the SRAM. I'm not quite sure how to manage that, and whether it's possible to read such large data files. I did some quick research and did not find what I wanted. It is likely possible, but let's do something simpler. Instead, I will use the EEPROM to store the last 5 minutes of data or so, and then have the server read the data every five minutes. This keeps the client simple, and puts the burden of complexity on the server. In addition, I don't want to read SD cards manually, as that is cumbersome, though possibly scripts could be made.

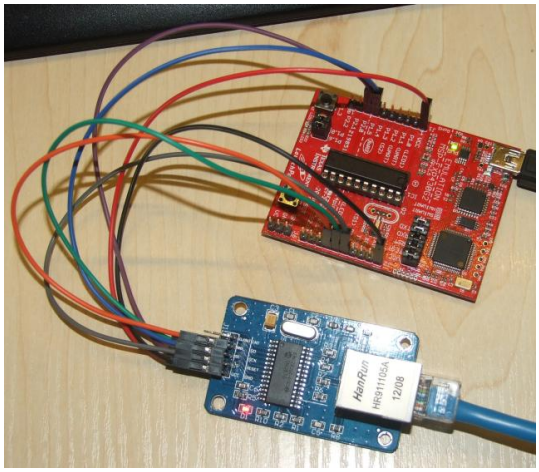




The PPD42 was made sure to be put vertically.

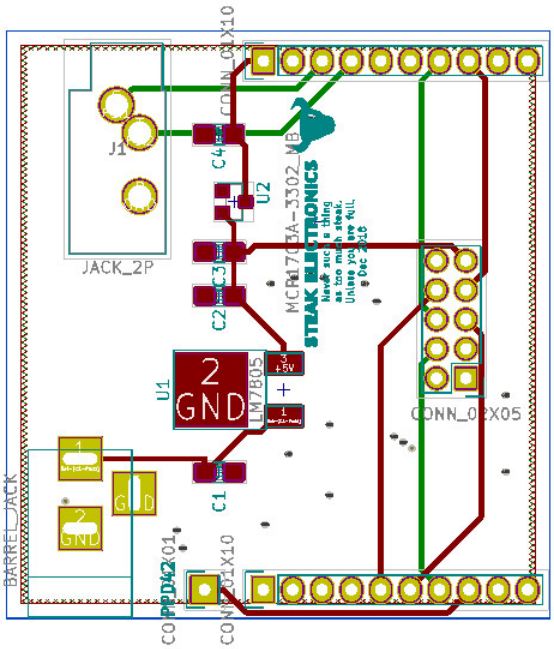
EDIT: 12/2018
I have decided to change how I do this slightly. Instead of an SD card, I will connect on the LAN and use thingspeak from a locally hosted instance (and deployed with docker, possibly) or some other aggregating server to pull the data. I'll also make a shield, to simplify deployment. That might not be necessary, but I can make a shield in a few hours, and pcbs are cheap. These options such as thingspeak sometimes have graphing / plotting included.

I'm going to use this library:
https://github.com/reaper7/EtherEncLib/releases
so git clone that, then
git tags -l
git checkout tags/v0.4.2
to get the latest release (or a newer one if possible).
The pinout for the ENC is viewable at the figure to the right. This is from the 43oh.com forum.



ENC and MSP430 (Note: Works with only two confirmed MSP models).

PCB Layout
layout is simple for the most part. I again, flipped the ENC as I had done on the Uno board, so it is inserted upside down on the board. This time I went only with the 2x5 pin enc for simplicity sake. The shield is below the MSP, so longer pin headers will be used (already have those) to give space for barrel plug. The board needs a 3.3v regulator and 5v for the PPD42. To get the sizing of the shield right, I aligned my grid with that of the design files for the msp430 dev board, and made sure the spacing between the 0.1” headers was exactly the same – easy. Ran all traces of 20 mils and made the board small as reasonably possible. Now to get them made, and actually test this.



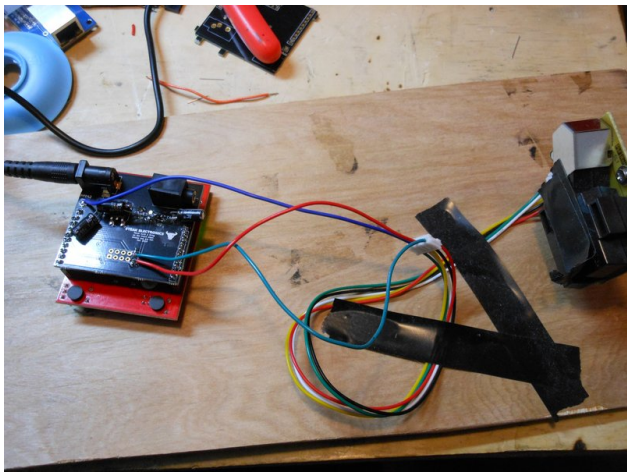
Spacing for the headers is board perfect, by aligning my grid, with the MSP430's design files grid (eagle), and making them align. FOSS+OSH win.

Layout + schem done in about 60-90 minutes. Easy.

Revision 1 Results:
I built up the PCB today. A few errors were made, but this is much better than the mess I had on the plywood originally. First off, the ENC pins are wrong, so that is not going to be able to be mounted (unless I wanted to wire pins manually). Second, the orientation of the MCP part is wrong. I'm not sure how this was done incorrectly, but I suspect that the default KiCad libraries were wrong. I have a few other changes to do, but this was overall successful, and I can respin another board quick quickly. On my todo list is:

- add gnd breakout
- add 5v breakout
- add 3v3 to enc NOT 5v
- fix enc pins
- fix orientation of main board
- add anime picture to back of board
- fix orientation of 3v3 reg
- buy more msps (need two more at least)
- buy 100 1uf 0805 caps put in main box (I'm surprised I don't keep these in stock. I do have through hole 1uf but not 0805).
- buylist + pcb : header so you can just plug in ppd. This is a sort of wish list thing, but something that will save a few seconds on build time. Although NOT necessary.

It looks like the connector between the board to the PPD42 is a type of 2.54 or 0.1” header. I will add a separate connector for the PPD42 that has all five pins.



Revision 1 Testing

BOM Creation Notes

The MSP-EXP430G2 is being obsoleted, in favor of the MSP-EXP430G2ET. This is not a good thing, and shows poor sport on Tis part. I was always wary of the two dozen MSP430's, and now they are removing the original one from production, according to digikey. Ugh. Makes me think I should've just used an Arduino, but I digress.

Planned obsolecense means, I'll have to make my own dev board for a TI part next time. Which I may just do, the chip is simple enough, and if ICSP and energia will work with this (need to check), then that is good enough.

Work Log 01/24/19

Adding Library to Energia
The path for adding a library to energia is:
/home/dev/Desktop/code/electronics/airqualitysensor/energia-0101E0012/hardware/msp430/libraries
on my machine. The idea is to add the library to the libraries folder (but, NOT the lib folder…, ugh). And you might have to hunt to find that. So I git cloned the ethernet lib, and let’s add it in. Remember that this ENC library works with only certain launchpads. Including, my now OBSOLETE one. F$#% companies, and their obsolecense. There’s a special place in hell for these people.

Upon adding the Ethernet libraries, and running websrv example, I get this error:
fatal error: avr/pgmspace.h: No such file or directory
compilation terminated.

After some fumbling, t looks like
https://github.com/energia/Energia/commit/cafa204a33e6e0dfa65b2f97dd14792a5964837e
is what I should be using.

Make sure you grab from the msp430 section, not the other cores… There’s only two header files, and one short c file. Then it’s all dupes for different cores (not sure if 100% alike, but we only want msp430).
Well this library is shit. Had to manually add an include to EtherEncLibUdp.cpp for the dtostrf.h file. I also had to put that file somewhere it could be found.
Fuck. 15 minutes of my life, I’ll never get back.

After that it builds.
I am able to see pings hitting the mac address, but no webserver response. Possibly it’s the CS pin that is wrong. I think it’s by default pin 10, but on the pin mappings, it should be 8. There’s two places to change this. One in the sketch (obvious). The second is in etherenclib somewhere… I saw this mentioned in the post in resources I have saved.

Ah, looks like I forgot to solder some pins. Woops. After that's out of the way, let's look at the pin mappings.
Quote: "(you can change this pin in file enc28typedef.h - line 424, then also mandatory! in sketch pinMode(10,OUTPUT);"
Says the forum post. I believe it's this:
#if (ENERGIA)

```
 #define ENC28J60_CONTROL_CS        8
 //#define SPI_MOSI                 13
 //#define SPI_MISO                 12
 //#define SPI_SCK                   14
```

Not sure why they think it's cute to obfuscate. Maybe it's an honest error. Not sure.

And with that 1 hour of my life, I was able to get this to work. Note that pings do NOT work. It should do nothing if you ping. If you are getting destination not found, then something is wrong. Pings simply reach a dead end. Now to incorporate this with the air quality sensor. Wew.

Basic Server test
Works.
Basic Sensor test
Works
Basic Sensor and Server test
Fails. Out of memory.

Let's see at what point, the basic combination of server and sensor fails… What is pushing it over the barrier… I can put math.h and go through setup – no problem…

Ok, it's this block of code:

```
if ((millis()-starttime) > sampletime_ms)//if the sample time == 30s
   {
      ratio = lowpulseoccupancy/(sampletime_ms*10.0);  // Integer percentage 0=>100
      concentration = 1.1*powf(ratio,3)-3.8*powf(ratio,2)+520*ratio+0.62; // using spec sheet curve
      Serial.print(lowpulseoccupancy);
      Serial.print(",");
      Serial.print(ratio);
      Serial.print(",");
      Serial.println(concentration);
      lowpulseoccupancy = 0;
      starttime = millis();
   }
```

Let's cut it down. Interestingly this code here, jumps up 4K bytes:

```
 if ((millis()-starttime) > sampletime_ms)//if the sample time == 30s
   {
      //ratio = lowpulseoccupancy/(sampletime_ms*10.0);  // Integer percentage 0=>100
      //concentration = 1.1*powf(ratio,3)-3.8*powf(ratio,2)+520*ratio+0.62; // using spec sheet curve
```

```
      Serial.print(lowpulseoccupancy);
      Serial.print(",");
      Serial.print(ratio);
      Serial.print(",");
      Serial.println(concentration);
      lowpulseoccupancy = 0;
      starttime = millis();
   },
```

even with ratio and concentration cut out. From about 10K to 14K bytes. Wow.

If I do this:

```
   if ((millis()-starttime) > sampletime_ms)//if the sample time == 30s
   {
      //ratio = lowpulseoccupancy/(sampletime_ms*10.0);  // Integer percentage 0=>100
      //concentration = 1.1*powf(ratio,3)-3.8*powf(ratio,2)+520*ratio+0.62; // using spec sheet curve
      Serial.print(lowpulseoccupancy);
      //Serial.print(",");
      //Serial.print(ratio);
      //Serial.print(",");
      //Serial.println(concentration);
      lowpulseoccupancy = 0;
      starttime = millis();
   }
```

It's back to about 10K bytes. So the serial prints, are a bit wild. I actually don't need serial prints at all. Let's cut those out.

```
ratio = lowpulseoccupancy/(sampletime_ms*10.0);  // Integer percentage 0=>100
```

this command is 2K bytes added.

The concentration post is simply too much, even by itself. The math is too expensive. It's 7K bytes about. There must be a cheaper way to do the math. Is there a pow alternative (pow gets the exponent, it appears)…
https://stackoverflow.com/questions/101439/the-most-efficient-way-to-implement-an-integer-based-power-function-powint-int
seems hopeful. Simple enough.

Now I have:

```
 if ((millis()-starttime) > sampletime_ms)//if the sample time == 30s
   {
      ratio = lowpulseoccupancy/(sampletime_ms*10.0);  // Integer percentage 0=>100
      //concentration = 1.1*powf(ratio,3)-3.8*powf(ratio,2)+520*ratio+0.62; // using spec sheet curve
      concentration = 1.1*ipow(ratio,3)-3.8*ipow(ratio,2)+520*ratio+0.62; // using spec sheet curve
      //Serial.print(lowpulseoccupancy);
      //Serial.print(",");
      //Serial.print(ratio);
      //Serial.print(",");
```

```
    //Serial.println(concentration);
    lowpulseoccupancy = 0;
    starttime = millis();
  }
```

Using a leaner integer pow, instead of a float exponent formula. Not sure how it will effect the data, but we will see. And it builds.

As far as the power dissipation of the linear vreg… I have a lm7805 on there, with a 12V regulator. In Troubleshooting Analog Circuits, Bob Pease mentions the 5 second rule for heat – if you can hold your finger (or thumb) on it for five seconds without being burned, it's about 85 deg C, and OK. If it's hotter, then you have issues. Well, I am right on the line. I could improve with a 7-9V AC Adapter, or a DC – DC switching regulator, but for now – this will do. I should mention, I'm touching the PCB opposite the LM7805 due to the construction. I should add more vias on the bottom of the vreg, also.

Work Log 1/25/19

Did some work tonight on sensor test. Looked at what the sensor was outputting (a HI signal with occasional LOW pulses), and decided to not do any math, but instead, just read the pulses. Please see Sensor test code for the progression of what I did. I've stripped out all the math, and only want relative values (i.e. has dust gone up or down).

I also did this on the web server code, now ethernet test fin. There is a eElib.print function, which is one of the cpp print functions… I needed higher resolution for the "sum" value which was a 32 bit integer, but the formula only has int (16 bit) covered. I've decided to simply shift the bits of sum, and lower the resolution. I don't want to edit the source code for this project. Although, I could add another function to cover my float use case. Oddly enough, the UDP code has a float print in place, but that doesn't help me.

This is the code that I am using:

```
// Print method
// Int values version of print method
void EtherEncLib::print(unsigned int val)
{
   if (DEBUGLIB) Serial.println(F("Printing2: "));
   char sI[] = { 0, 0, 0, 0, 0 };
   itoa(val, sI, 10);
   print(sI);
   if (DEBUGLIB) Serial.println();
}
```

It would be trivial to add another function, but not as simple as I'd like.

One thing I should do is test the pulseIn third variable – the one that measures how long to wait for

pulses to start. I can lower it, and get more readings, possibly.