Hardware Overview

The hardware in the project is broadly divided into:

- Sensors and their interfacing with Arduino Nano
- Arduino Nano and NodeMCU interface

In this project, the Figaro TGS2600 checks for Methane and Carbon Monoxide, Figaro TGS2602 checks for Ammonia, Hydrogen Sulfide and Toluene while the less accurate and cheap MQ135 checks for Carbon Dioxide and NOx. Although a single sensor is used to detect multiple gases, it's better to use a dedicated sensor for each gas, if you want better accuracy and reliability.

Arduino Nano and NodeMCU should cost less than about $15 in total and will do a great job for our basic requirement.

Sensors: Interfacing and Testing with Nano

Most low cost Gas Sensors available in the market are Electrochemical-type contact sensors. A sensing element is used to detect a certain gas. The sensor conductivity increases/decreases depending on gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration To model this, it's extremely important to understand the datasheets and the Rs/Ro Vs PPM or the Sensitivity Characteristics.

For TGS2600/TGS2602 Sensor interface with Arduino

The sensor requires two voltage inputs to detect the concentration of gas: Heater Voltage (VH) and Circuit Voltage (VC). This sensor has 4 terminals. Two are dedicated to heater which maintains optimum sensing temperature in the sensor and the other two are for measurement of Output Voltage across a load resistor (RL) which is connected in series with the sensor. DC voltage is required for the circuit voltage since the sensor has a polarity. A common power supply circuit can be used for both VC and VH to fulfill the sensor's electrical requirements.

The value of the load resistor (RL) should be chosen to optimize the alarm threshold value, keeping power consumption (PS) of the semiconductor below a limit of 15mW. Power consumption (PS) will be highest when the value of Rs is equal to RL on exposure to gas (source: TGS2600 Datasheet).

From the sensitivity curve, we obtain the scaling factor (a), and exponent (b) for our equation the ppm = a*(Rs/Ro)^b using power regression. If you don't understand this, please refer to Davide's Blog.

Also,Rs = Sensor resistance in displayed gases at various concentrations Ro = Sensor resistance in fresh air

To calculate Ro:

We know what the values of a, b are for 'air' (which is typically constant throughout) and also the Rs value, which can be calculated from:
Rs = (Vc/Vout - 1) x RL

Hence from the equation: ppm = a*(Rs/Ro)^b we get:
Ro = Rs * sqrt(a/ppm, b)

Once you obtain the Ro value, you can now proceed to calculate the final PPM value. To do so, follow these steps:

- Convert raw analog value to Vout or Vrl Voltage value
- Calculate the Rs value
- Find Rs/Ro ratio
- Obtain the PPM value by plugging the obtained values into the formula

This should help you obtain gas sensor value in PPM. However, as a part of calibration, you may be required to use a true value as reference and compare against the measure value.

Use: Ref. Value = Calibration Factor x Measured Value

Arduino Nano and NodeMCU interface

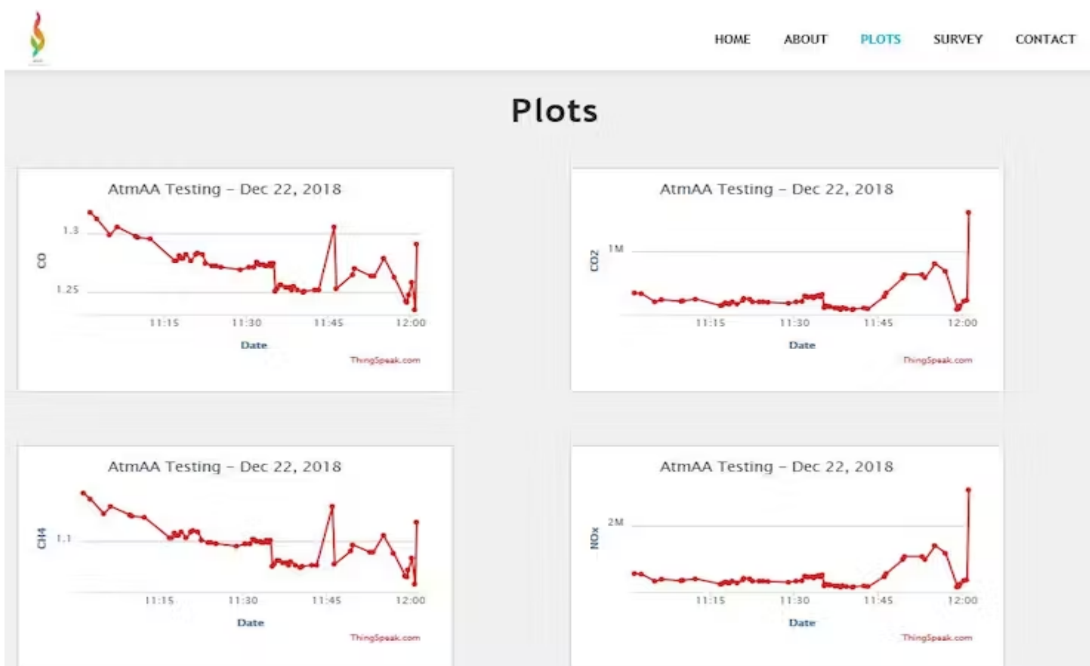
While hardware for sensors and their connectivity with Arduino Nano is not quite surprising and is fairly common, the Interface between Arduino Nano and NodeMCU might be intriguing to a few. The objective here was to ensure that the device be cheap and should have connectivity to the internet. The necessity of Arduino Nano arises when data from sensors is to be sampled. The gas sensors output Analog data and hence the MCU must have at least 3 analog pins (since we used 3 gas sensors here; could be scaled up to a larger number) to read this analog data.

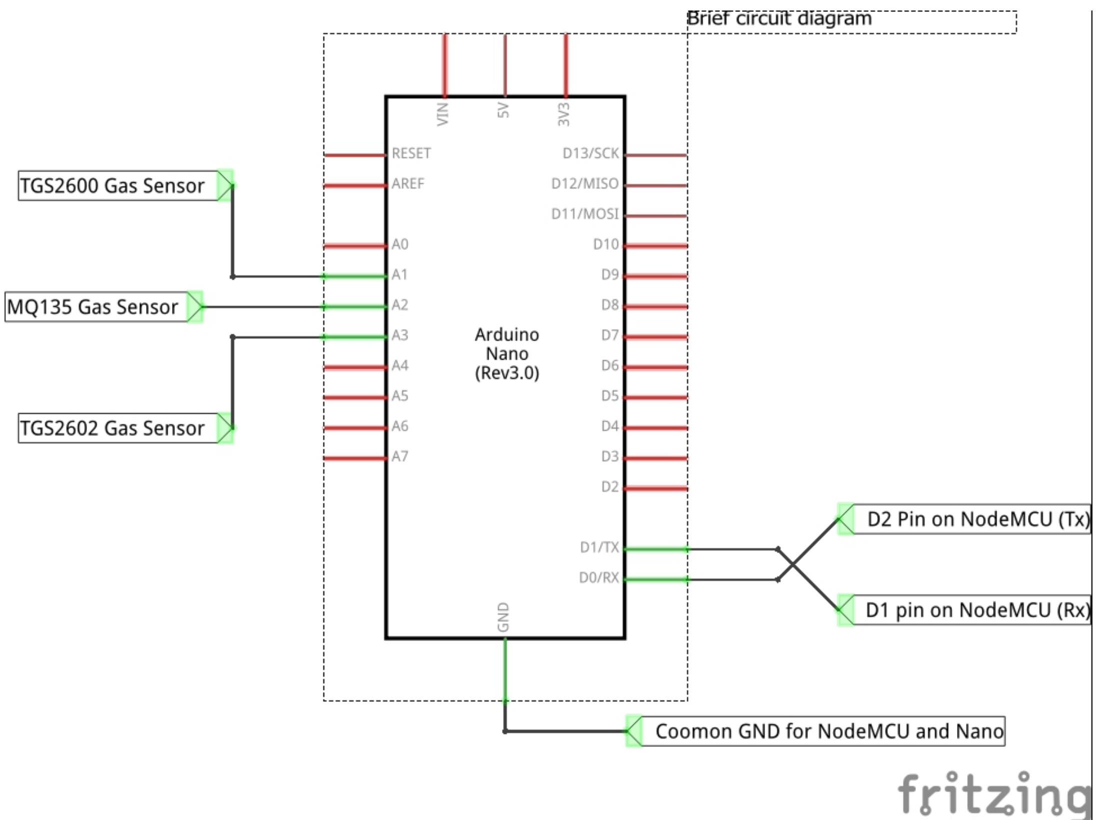
As NodeMCU has only 1 Analog input pin (A0), it makes it alone isn't fit for the job. To solve this, the most obvious approach would be to interface an ADC (Analog-to-digital) converter and get this job done, but I chose a rather uncommon approach by interfacing a NodeMCU and a Nano (which was interfaced to the sensors). I did this because both these devices commonly available and easy to use as compared to an ADC, thus making it convenient.

To interface Nano and NodeMCU is quite simple. We make use of the "SoftwareSerial.h" library, which allows us to use any set of digital pins on the MCU as our UART bus. We directly used Nano's default Rx/Tx pins and cross-connected them with pin 6 and 5 respectively on NodeMCU to establish UART communication.

Rx (Nano) <=> Tx (Digital pin 6 on NodeMCU)
Tx (Nano) <=> Rx (Digital pin 5 on NodeMCU)
GND (Nano) <=> GND (NodeMCU)

While testing the UART communication, I realized that the entire data wasn't getting transferred and I couldn't debug it either. Hence, I decided to use the standard JSON format to transfer data. With the help of "ArduinoJson.h" library, I could pack this data, send it, store it on a buffer till the entire data is received, unpack it and then upload it to ThingSpeak. To understand how to implement Arduino JSON to your project, head over to the creator's website or read the documentation of the library on GitHub.


HOME
ABOUT
PLOTS
SURVEY
CONTACT
Plots
AtmAA Testing – Dec 22, 2018
CO
1.3
1.25
11:15
11:30
11:45
12:00
Date
ThingSpeak.com
AtmAA Testing – Dec 22, 2018
CO2
1M
Date
ThingSpeak.com
AtmAA Testing – Dec 22, 2018
CH4
1.1
Date
ThingSpeak.com
AtmAA Testing – Dec 22, 2018
NOx
2M
Date
ThingSpeak.com


Brief circuit diagram
VIN
5V
3V3
RESET
AREF
A0
A1
A2
A3
A4
A5
A6
A7
D13/SCK
D12/MISO
D11/MOSI
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1/TX
D0/RX
GND
Arduino
Nano
(Rev3.0)
TGS2600 Gas Sensor
MQ135 Gas Sensor
TGS2602 Gas Sensor
D2 Pin on NodeMCU (Tx)
D1 pin on NodeMCU (Rx)
Coomon GND for NodeMCU and Nano
fritzing