1-1-2013

Characterization of MQ-Series Gas Sensor Behavior

Alec Fisher

Northern Illinois University

Capstone Project Report

Characterization of MQ-Series Gas Sensor Behavior

Author:
Alec Fisher

Faculty Supervisors:
Dr. Martin Kocanda
Dr. David Ballantine

Acknowledgments

I would like to acknowledge Dr. Martin Kocanda of the departments of electrical engineering and chemistry and Dr. David Ballantine of the department of chemistry. Nothing I did would have been possible without their help. Dr. Haji-Sheikh of the department of electrical engineering contributed helpful insight regarding the nature of semiconductors that was instrumental in explaining certain data trends. I would also like to thank Country Gas of Crystal Lake, IL for generously donating a propane cylinder and fittings to aid my research. Furthermore, the machine shops in both the engineering building and John E. La Tourette Hall were tremendously helpful in the process of constructing my testing apparatus.

Abstract

There are a wide variety of circumstances in which knowing of the presence of various potentially toxic gases in the local atmosphere is desirable. For many cases, such as for average household use, inexpensive alarms or detectors are readily available for specific gases. Common examples of these are carbon monoxide detectors and smoke alarms. However, for some applications, a higher degree of precision and detail about the properties of the local environment is desired. Unfortunately, commercial solutions to this problem are prohibitively expensive for many cases. The purpose of this project was to study the behavior of MQ-series gas sensors. These are metal oxide semiconductor gas sensors which are inexpensive and readily available from internet retailers which makes them seemingly ideal for the entrepreneur or hobbyist looking to work with gas sensors. However, the sensors are not internally calibrated, so characterizing their behavior is necessary before they can used to collect quantitative data. While there are others available, the models tested for this experiment were the MQ2 (flammable gas and smoke), MQ4 (methane), MQ6 (isobutane/propane/LPG), MQ7 (carbon monoxide), and the MQ9 (carbon monoxide/flammable gas). The primary goal was to find a relationship describing the sensor signal as a function of gas concentration and the sensitivity resistance. To accomplish this, a testing apparatus was created such that a controlled mixture of a gas to which the sensor is sensitive (propane, referred to generally as the "active" gas) and an inert gas (nitrogen, generally referred to as the "inactive" gas) could be passed over the sensor surfaces. Different combinations of active gas concentration and sensitivity resistor values were explored to develop the desired functional relationship. Upon examination of the results of over one-hundred trials, it was concluded (in accordance with equations derived analytically) that there was a strong relationship between sensor value and active gas concentration resembling a logistic function. Additionally, it was found that the resistance across the sensor itself is strongly dependent on the voltage drop across it and its reference atmosphere, though the nature of said relationships were not determined due to the multitude of potentially influential factors regarding which either little information was available or studying would be prohibitively difficult.

Experimental Setup

The experimental setup was designed such that a gas mixture with known proportions of each gas could be continuously passed over the sensors. To create the gas mixture, there were two gas sources: a cylinder of nitrogen and a cylinder of propane. The sensors are responsive to the presence of propane and not to that of nitrogen. The outlets of each gas cylinder lead through pressure and flow regulators and into a mixing chamber which lead to the sensor apparatus. The pressure regulators were necessary to stabilize the flow and the flow regulators were used to vary the concentration of the active gas between trials. The apparatus itself was a sealed plastic box with inlets and outlets for gas and holes for the sensors. To prevent losses, each hole and some of the tube junctions were sealed with silicone sealant. The entire setup (with the exception of the gas cylinders) was contained within a fume hood for safety purposes. The sensor apparatus is shown below in figure 1.

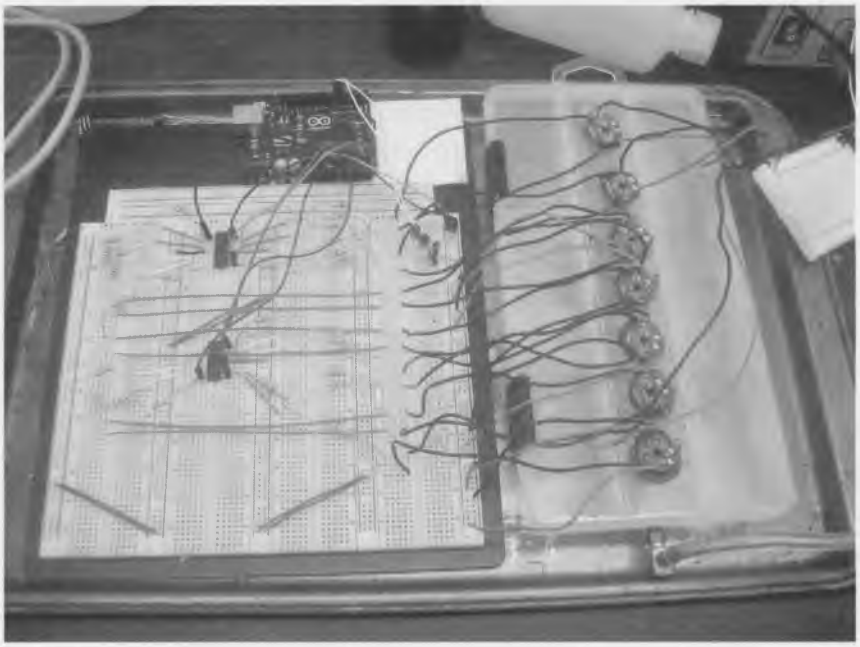


Figure 1: Sensor apparatus

The red discs shown are the breakout boards that were used for the sensors (small circuit boards that simplify integrating them into some larger system). The specifics of the sensors will be discussed in further detail later. Power was supplied by an Elenco 620-XP power supply that was built from a kit before the start of the semester. The signals from each sensor were sent to the Arduino Uno microcontroller's analog input pins where they were written to a text file on an SD card. Upon completing a test, the SD card was removed from the Arduino SD card shield and read by a laptop computer. The Arduino code for this can be found in Appendix A. The text file was formatted such that it could be directly imported into MATLAB for analysis. A MATLAB script was made to expedite the process of graphing the data for initial viewing and several

other functions and scripts were used in the process of analysis. This script along with other MATLAB code which was written to analyze and process the data can be found in Appendix B. Before another test could be run, the contents of the data file needed be deleted, so the file must have been copied and stored on the computer after each test.

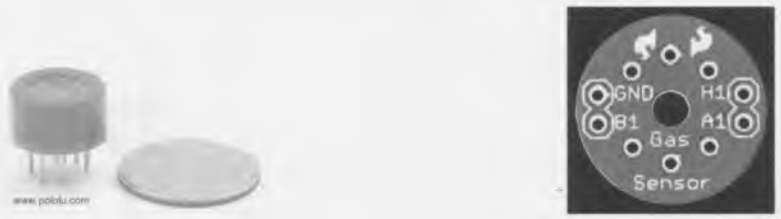
Sensors

Theory

The sensors are all metal oxide semiconductor (MOS) gas sensors. Since they are semiconductors, they can be either n-type or p-type (generally speaking). In n-type sensors, the charge carriers are electrons. Conversely, the charge carriers in p-type sensors are positive holes. The presence of an oxidizing or reducing gas changes the resistance of the film by changing the surface charge carrier concentration via altering the amount of oxygen adsorbed on the surface. This change in charge carrier concentration is typically effected by the adsorption of oxygen on the film surface. Due to their opposite effects on charge carrier concentration, a reducing or oxidizing gas will have opposite effects on the resistance of the sensors. Since reducing gases increase the number of electrons on the surface, they lower the resistance in n-type semiconductors, but increase the resistance in p-type semiconductors. Similarly, since the presence of an oxidizing gas will reduce the number of electrons on the surface, it will increase the resistance across n-type sensors and decrease the resistance across p-type semiconductors[1]. The sensors studied in this experiment use a thin film of tin oxide (SnO_2), an n-type semiconductor.

Construction

Each sensor has six pins: two A pins, two B pins, H, and GND. The breakout board takes these six pins and outputs to four pins: A, B, H, and GND. Both a sensor and breakout board are shown below in figure 2a and figure 2b respectively.



(a) MQ9 sensor

(b) Breakout Board

Figure 2: Breakout board for MQ-series sensors (left) and an MQ 9 sensor

Pins H and GND are the pins that power the internal heating coil; since the resistance across the coils vary with temperature, maintaining the sensor surface at a high temperature has been found to help improve the consistency of results. On the sensor, the A pins are on the opposite side of the MOS thin film from the B pins; the breakout board only uses pins one each of the A and B pins. To use the sensor, a 5V (AC or DC) source is connected across the heater coil pins to heat up the sensor. Additionally, pin A is wired to the 5V source in such a manner that it induces a current across the sensor surface and out pin B, where the current is then routed through a resistor to ground. The wire that outputs the signal to the microcontroller is connected between the sensor and the resistor to form a voltage divider circuit. By adjusting the resistor, the sensitivity of the sensor can be altered. Because of this, the resistor in this circuit is often referred to as the sensitivity resistor. A circuit diagram depicting the circuit described above is

shown in figure 3. Both heater coil pins are labeled "H" because choosing which pin is on the positive or negative side of the voltage source is arbitrary. The purpose of the heater coil is to maintain the sensor at an elevated temperature, as the sensor response has been found to be sensitive to temperature.

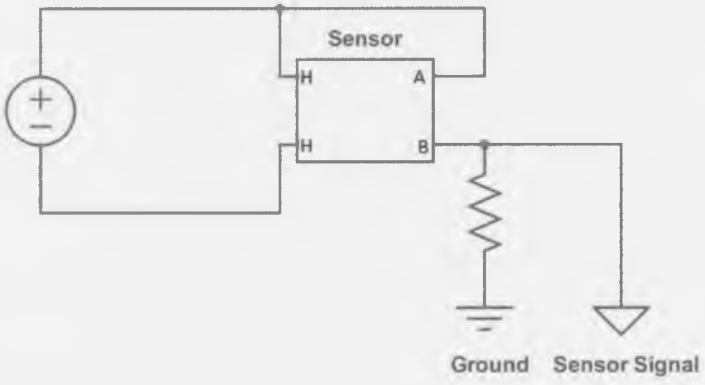


Figure 3: Circuit for applying sensors

Quantitative Results

Signal readings were taken at ten different resistor sensitivities and at ten different gas concentrations (one-hundred readings per sensor, totaling seven-hundred readings). The data was collected under similar temperatures each time to minimize environmental variance between trials. A typical plot of signal response as a function of gas concentration is shown in figure 4.

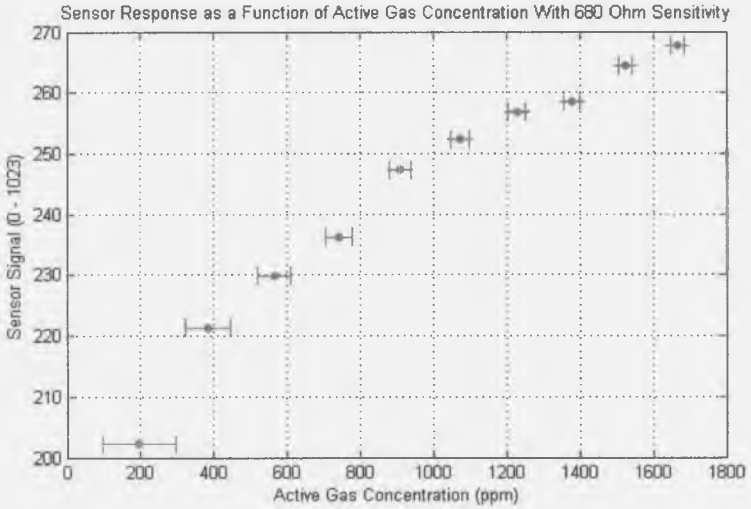


Figure 4: Sensor response as a function of gas concentration with the sensitivity resistor set at 680Ω.

Given that the sensor circuit is a simple voltage divider circuit (whose measurand is the voltage drop across the sensitivity resistor scaled from 0V - 5V to an integer from 0 - 1023 by the microcontroller), application of Ohm's law dictates that the voltage drop can be calculated as shown in equation 1 where V_s is the voltage drop across the sensitivity resistor with resistance R_s and V_{max} is the voltage of the source (the maximum voltage drop across the sensitivity resistor as the resistance of the sensor, R_v, approaches zero).

$$V_s = V_{max} \frac{R_s}{R_s + R_v} \tag{1}$$

Since V_{max} and R_s are constant for any given trial, it is clear that R_v must change as a function of gas concentration in order to effect changes in V_s. By calculating the voltage drop across the sensor by subtracting the resistor voltage from the source voltage and dividing that voltage drop by the current (obtained by dividing the voltage drop across the resistor by its resistance), a plot of data for the resistance of the sensor as a function of gas concentration was obtained. Such a plot for the same parameters as the previous figure is shown in figure 5.

This figure suggests that the sensor has some initial, maximum value when no active gas is present (referred to later as R_0). With increasing exposure to the gas, the sensor resistance decays exponentially, asymptotically approaching some minimum value once

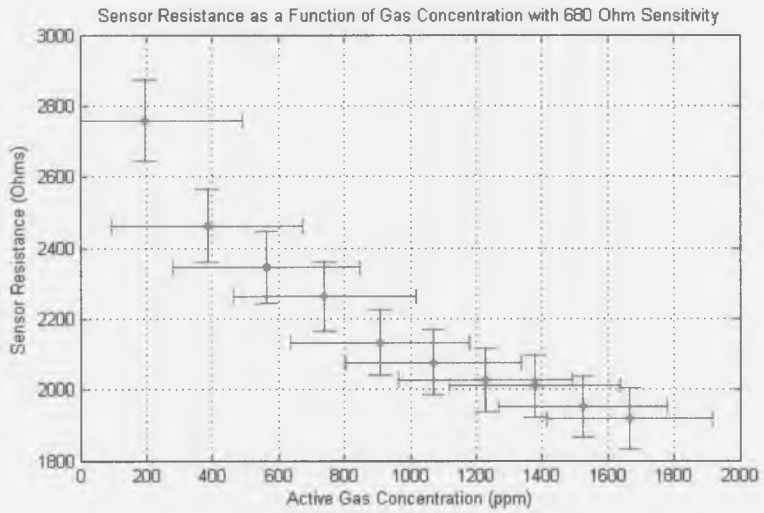


Figure 5: Sensor resistance as a function of gas concentration

the sensor has been saturated (referred to later as R_∞). This theoretical relationship is shown in equation 2 where x represents the gas concentration.

$$R_v = R_\infty + (R_0 - R_\infty)e^{-\beta x} \tag{2}$$

It logically follows that, if a curve of best fit can be found to calculate R_v based on these parameters, it can be easily substituted into equation 1 to obtain a function to calculate the voltage across the sensitivity resistor (the measurand for the circuit) as a function of active gas concentration. To do this, the method of least squares was implemented. The method of least squares quantifies error for one data point to be the square of the difference between the observed value and the predicted value from the best fit curve and the error for a data series to be the sum of the square of the differences between each point in the data series and the best-fit curve.. Mathematically speaking, for an arbitrary best fit curve $y = f(x)$ with observed coordinate pair (x_0, y_0), the least-squares error for one point and for an entire data set are seen in equations 3 and 4.

$$S = (y_0 - f(x_0))^2 \tag{3}$$

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{4}$$

As one would do to find the minima in any function, the proper course of action now is to take the derivative of $f(x)$ with regard to the parameter being solved for (in this case, β) and set it equal to zero. The sum and the derivative are both linear operators, so the order in which they are applied is arbitrary (the derivative of the sum is equal to the sum of the derivatives). This process for the equations derived above is shown below. For readability and ease of understanding, the following substitution has been made after the statement of the initial equation:

$$u_i = R_s + R_\infty + (R_0 - R_\infty)e^{-\beta x_i} \tag{5}$$

$$S = \sum_{i=1}^{n} \left(V_{si} - \frac{V_{max} R_s}{R_s + R_\infty + (R_0 - R_\infty)e^{-\beta xi}} \right)^2 \tag{6}$$

$$S = \sum_{i=1}^{n} V_{si}^2 - \frac{2V_{si}R_s V_{max}}{u_i} + \frac{V_{max}^2 R_s^2}{u_i^2} \tag{7}$$

$$\frac{dS}{d\beta} = \sum_{i=1}^{n} \frac{2V_{max}V_{si}R_s}{u_i^2}\frac{du_i}{d\beta} - \frac{2V_{max}V_{si}^2 R_s^2 u_i}{u_i^4}\frac{du_i}{d\beta} \tag{8}$$

$$0 = \sum_{i=1}^{n} V_{si} - \frac{V_{max}R_s}{u_i} \tag{9}$$

$$\sum_{i=1}^{n} V_{si}(R_s + R_\infty + (R_0 - R_\infty)e^{-\beta x_i}) = \sum_{i=1}^{n} V_{max}R_s \tag{10}$$

$$\sum_{i=1}^{n} V_{si}(R_0 - R_\infty)e^{-\beta x_i} = \sum_{i=1}^{n} V_{max}R_s - V_{si}R_s - V_{si}R_\infty \tag{11}$$

$$\sum_{i=1}^{n} e^{-\beta x_i} = \sum_{i=1}^{n} \frac{V_{max}R_s - V_{si}R_s - V_{si}R_\infty}{V_{si}(R_0 - R_\infty)} \tag{12}$$

$$\sum_{i=1}^{n} -\beta x_i = \sum_{i=1}^{n} \ln \frac{V_{max}R_s - V_{si}R_s - V_{si}R_\infty}{V_{si}(R_0 - R_\infty)} \tag{13}$$

$$n\beta = \sum_{i=1}^{n} \frac{-1}{x_i} \ln \frac{V_{max}R_s - V_{si}R_s - V_{si}R_\infty}{V_{si}(R_0 - R_\infty)} \tag{14}$$

$$\beta = \frac{1}{n} \sum_{i=1}^{n} \frac{-1}{x_i} \ln \frac{V_{max}R_s - V_{si}R_s - V_{si}R_\infty}{V_{si}(R_0 - R_\infty)} \tag{15}$$

Upon examination of the data, it was found that one of the initial assumptions regarding the nature of the system - that, regardless of the voltage drop across the sensor, its resistance would start at the same R_0 and asymptotically approach the same R_∞ - was incorrect. The resistance as a function of concentration for the same sensor under different voltage drops (the result of using different values for the sensitivity resistor) is shown in figure 6. Though not technically appropriate, the data points in each data series are connected with lines to distinguish between them.

The relationship between the voltage drop across the sensor (which is directly related to the value of the sensitivity resistor) and the sensor resistance was not clearly understood. This relationship is shown for different gas concentrations in figure 7. Again, while not technically appropriate, the points in each data set are connected with lines to indicate distinct data sets. Each data set represents the resistance of the sensor for the same gas concentration as the voltage drop across it is varied.

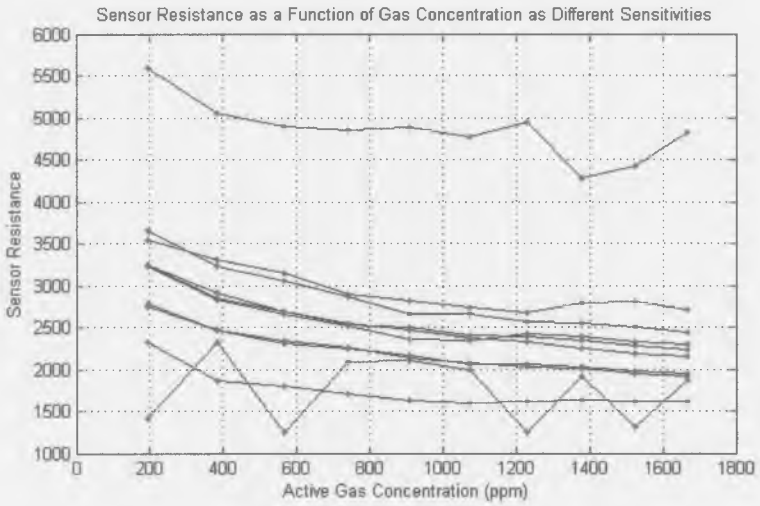


Figure 6: Sensor resistance as a function of gas concentration at different levels of resistor sensitivity

Considering that the behavior of semiconductors is not within the realm of expertise for the average mechanical engineer, consultation was necessary. A brief discussion with Dr. Haji-Sheikh, who was known to have knowledge of semiconductor theory, revealed that such dependencies are not uncommon in polycrystalline semiconductor films; since electric mobility and the conditions for quantum tunneling between grain boundaries are influenced by the electric field across the sensor (which is the result of a potential difference across the surface), changing the voltage can significantly affect the resistivity (and therefore resistance) of the material. However, based on what data is available, attempting to analytically derive a relationship would be impossible. So, rather than finding a way to solve for R_0 and R_∞ for any desired R_s, they were simply estimated for each sensitivity resistor setting (the discovery that they were not equal for all conditions occurred after the testing apparatus had been dismantled). If this procedure for characterizing sensor behavior were repeated with another sensor, these values would simply be measured, negating any need for approximation. While approximate, the estimated values yielded rather accurate results for the curves of best fit. An example is shown in figure 8.

Now that a best-fit curve equation is known, all that remained was rearranging it such that the concentration could be found as a function of the sensor signal. The rearranged equation is shown below.

$$x = \frac{-1}{\beta} \ln \frac{V_{max} R_s - V_{si} R_s - V_{si} R_\infty}{V_{si}(R_0 - R_\infty)} \tag{16}$$

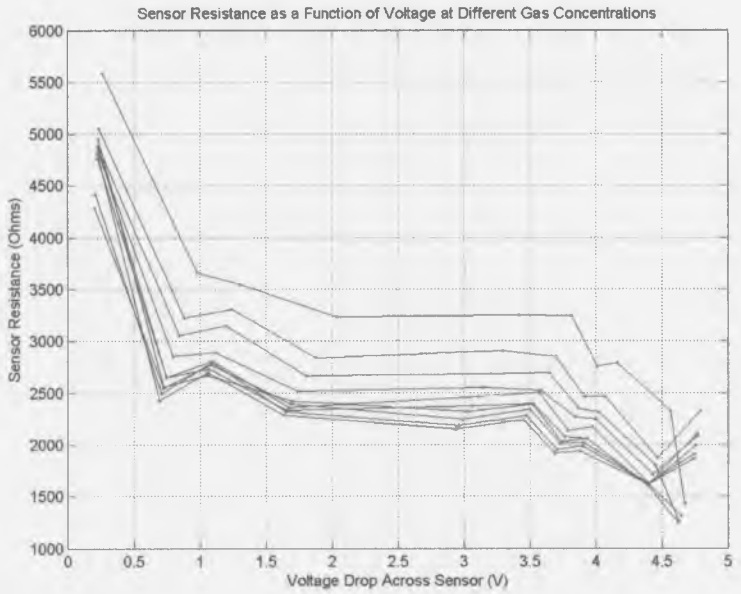


Figure 7: Variance in sensor resistance as a function of voltage at different gas concentrations

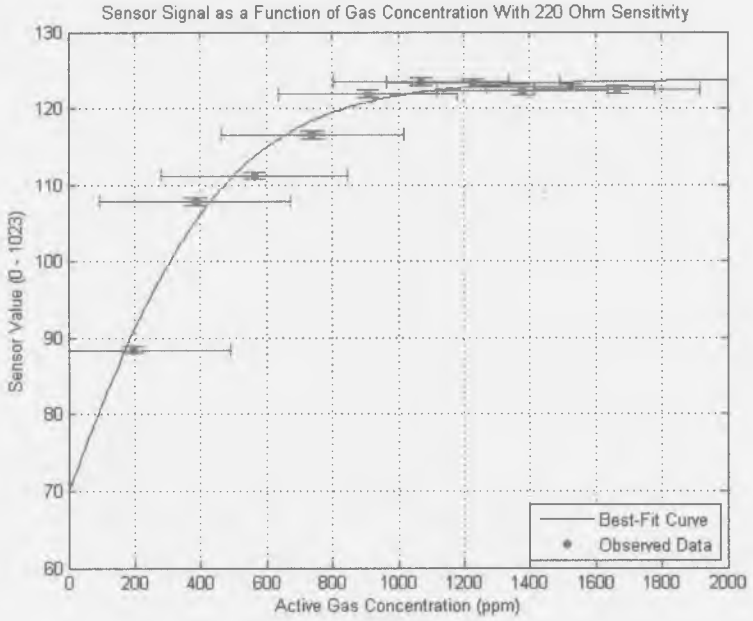


Figure 8: Sensor response as a function of gas concentration with superimposed best-fit curve

Uncertainty Analysis

With regards to error in accordance with the standard procedures of propagating uncertainty, it was found that the following equation represents the uncertainty in the concentration for any set of parameters when calculating it as a function of sensor signal is as follows in equation 17.

$$u_x^2 = \left(\frac{\delta x}{\delta R_s}\right)^2 u_{Rs}^2 + \left(\frac{\delta x}{\delta R_\infty}\right)^2 u_{R_\infty}^2 + \left(\frac{\delta x}{\delta R_0}\right)^2 u_{R_0}^2 + \left(\frac{\delta x}{\delta V_s}\right)^2 u_{V_s}^2 + 2\left(\frac{\delta x}{\delta R_0}\right)\left(\frac{\delta x}{\delta R_\infty}\right) u_{R_s} u_{R_\infty} \tag{17}$$

Where

$$\frac{\delta x}{\delta R_s} = \frac{-1}{\beta}\left(\frac{V_{max} - V_s}{V_{max}R_s - V_sR_s - V_sR_\infty}\right) \tag{18}$$

$$\frac{\delta x}{\delta R_0} = \frac{1}{\beta(R_0 - R_\infty)} \tag{19}$$

$$\frac{\delta x}{\delta R_\infty} = \frac{1}{\beta}\left(\frac{V_s^2(R_0 - R_\infty) - V_s(V_{max}R_s - V_sR_s - V_sR_\infty)}{V_s(R_0 - R_\infty)}\right) \tag{20}$$

$$\frac{\delta x}{\delta V_s} = \frac{-1}{\beta}\left(\frac{V_s(R_s - R_\infty) + (V_{max}R_s - V_sR_s - V_sR_\infty)}{V_s(R_0 - R_\infty)(V_{max}R_s - V_sR_s - V_sR_\infty)}\right) \tag{21}$$

$$u_{R_0} = u_{R_\infty} = \left(\frac{V_{max} - V_s}{V_s}\right)^2 u_{R_s}^2 + \left(\frac{R_sV_{max}}{V_s}\right)^2 u_{V_s}^2 \tag{22}$$

$$u_{R_s} = .05R_s \tag{23}$$

$$u_{V_s} = \frac{15}{1023} \tag{24}$$

The most noteworthy part of these equations is the frequency with which R_s appears in the numerator. Given that values for R_s ranged from 100Ω at the least to 100,000Ω at most, the calculated error (almost entirely due to error in the resistors used, since its uncertainty is 5% of its given value, which is also very high for large resistors) was astronomically high: so high that the error bars would not appear on any of the charts at any meaningful scale. It is for this reason that error bars are only shown for observed data points whose error is relatively small and can be seen with ease.

Another source of error that was investigated was that of so-called "preheat error;" error resulting from the sensor not being at the proper temperature. The datasheets for all of the sensors tested listed a preheat time of 24-48 hours. Since it was not clearly stated, it was not known at the onset whether this was a one-time issue or if the sensors would always need to be powered continuously for that long before valid readings could be taken. The results of a long-term test aimed at characterizing preheat error after each sensor had been powered on and off several times are shown below in figure 9. Based on this figure, it is clear that the sensors reached their steady-state value in a relatively short amount of time (approximately ten minutes for most sensors).

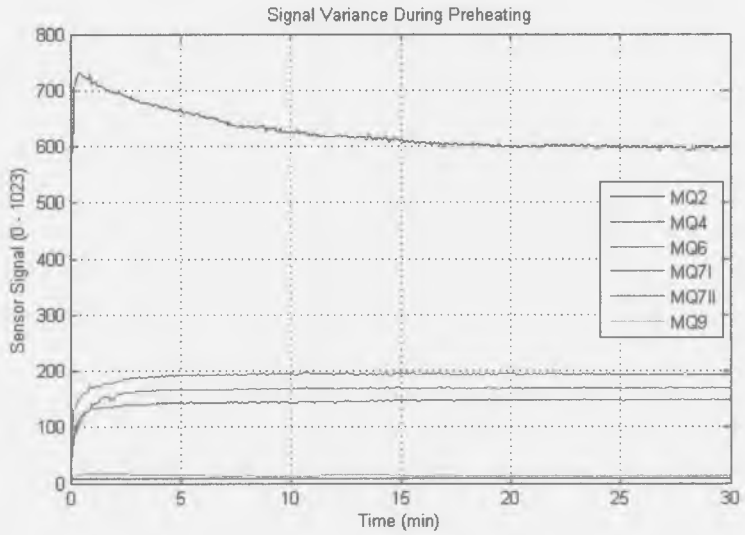


Figure 9: Variance in the sensor signal as the sensors heated up from room temperature to operating temperature

Qualitative Results

As shown in figure 10, it is clear to see that, while two of the sensors were the same model (MQ7 carbon monoxide sensors), they reported considerably different values. While it is possible that this is an anomaly, this suggests that there may be a great inconsistency between instances of the same model in general. Assuming this is the case, any sensors used in such applications will need to be individually calibrated to guarantee accuracy.

Another crucial fact that was has yet to be discussed is that the sensors must be calibrated with the inactive gas as that of the environment in which it will being used and to which all measurements are relative. All of the tests for this study were performed with nitrogen as the inactive gas because it was known to be a generally inert gas and it was readily available and controllable. This allowed for the sensors' behavior to be attributed to the propane. However, for use in a typical atmosphere, it should be calibrated with air. Since there is a significant amount of oxygen in the air, it has a significant oxidizing effect on the sensor, causing a dramatic increase in its resistance. While no trials of varied concentration were taken due to the inability to properly regulate air flow, a measurement from one of the sensors only in the presence of air resulted in a sensor resistance of nearly $600,000\Omega$: a huge value in comparison to the relatively modest values from $3,000\Omega$ to $4,000\Omega$ when immersed in nitrogen. Therefore, perhaps a more accurate definition of the R_0 used above numerous times in equations is the resistance of the sensor when it is fully saturated by its reference environment (e.g. air). Therefore, the values found for the least-squares curve equations with nitrogen as its reference environment are not valid when air is its reference environment; in order to use air for the reference environment, a series of tests would need to be done with varying mixtures of air and the gas of interest for the sensor (e.g. LPG, carbon monoxide, etc.). If the same values for the equation parameters (β, R_0, R_∞) are used, the results will be highly erroneous. However, the

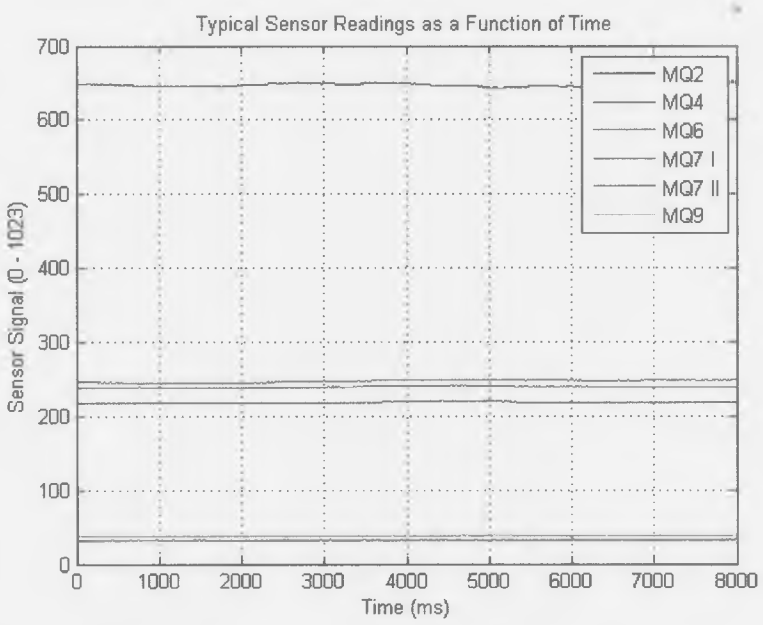


Figure 10: The results of a typical trial where the sensors were exposed to a particular gas mixture with a certain sensitivity resistor in place

procedure and equations described above still apply to any sensors of this type as long as the correct values for the parameters are used.

A more problematic finding was the potentially limited selectivity of the sensors. The selectivity of a sensor reflects how exclusively its target gas influences its readings. A sensor with a low selectivity might be strongly influenced by many gases in addition to that it was designed to detect. Since only one gas was extensively tested with the sensors in this experiment, no strong conclusions can be drawn generally speaking regarding the selectivity of any of the sensors. However, several of the sensors which were not advertised as detecting propane (such as the MQ7 carbon monoxide and MQ9 carbon monoxide/flammable gas sensors) were sensitive to it. While there is no guarantee that testing with other gases would reveal further issues with selectivity, it seems more likely than not that problems with selectivity could be significant obstacles to overcome. However, a tentative solution to this has been considered. Supposing that the goal is to accurately determine the concentrations of three different gases of interest and there exist three sensors that are at least somewhat sensitive to each of those gases, the equations found using the procedure above to characterize each sensor's response to each gas individually could be combined into a system of equations. Though this would need to be verified experimentally, it seems to be a potential solution to this problem.

A more expensive and complicated (though more effective) way would be to add coatings to the sensor surfaces to alter their selectivities. Research by many researchers has already confirmed that adding certain coatings can alter the selectivity toward some gases. Indeed, each of the sensors tested here probably has one or more coatings; since they all are made from the same metal oxide, coatings are the most likely explanation

for why some sensors were sensitive to propane and some were not. However, such information has not been made publicly available by the manufacturer.

References

[1] George F. Fine, Leon M. Cavanagh, Ayo Afonja and Russell Binions, *Metal Oxide Semi-Conductor Gas Sensors in Environmental Monitoring.* Sensors 2010, 10, 5469-5502; doi:10.3390/s100605469

Appendix A: Arduino Code

The following code is that which was used to log the data which was collected over the course of this experiment.

```
   commentstyle
#include <SD.h>   //includes the SD library

int gLEDpin = 4;     //declares green, red, and yellow LED pins
int rLEDpin = 3;
int yLEDpin = 2;
int time = 0;        //declares a variable for time

int MQ7II = A0;       //declares each sensor as an analog pin
int MQ9 = A1;
int MQ7I = A2;
int MQ4 = A3;
int MQ2 = A4;
int MQ6 = A5;

int x2;
int x4;
int x6;
int x7I;
int x7II;
int x9;
int y2;
int y4;
int y6;
int y7I;
int y7II;
int y9;

int flag = 0;        //declares a flag variable as 0 (explained later)
int inc = 2500;         //sets the time interval (milliseconds)

void setup()
{
  Serial.begin(9600);
  SD.begin();                  //intiates the SD card
  pinMode(gLEDpin,OUTPUT);     //declares LED pins as outputs
  pinMode(rLEDpin,OUTPUT);
  pinMode(yLEDpin,OUTPUT);

  int temp2[10] = {0,0,0,0,0,0,0,0,0,0};   //Everything from here to the loop is
  int temp4[10] = {0,0,0,0,0,0,0,0,0,0};   //a test to check if the readings have
  int temp6[10] = {0,0,0,0,0,0,0,0,0,0};   //reached a near-steady-state value
  int temp7I[10] = {0,0,0,0,0,0,0,0,0,0};
  int temp7II[10] = {0,0,0,0,0,0,0,0,0,0}; //Once it does reach a steady state,
  int temp9[10] = {0,0,0,0,0,0,0,0,0,0};   //it begins recording data
  int flag = 0;
  int ind = 1;
  int flag2 = 0;
  while(flag==0)
  {
    digitalWrite(yLEDpin,HIGH);
```

```
digitalWrite(gLEDpin,LOW);
digitalWrite(rLEDpin,LOW);

temp2[ind] = analogRead(MQ2);
temp4[ind]  = analogRead(MQ4);
temp6[ind] = analogRead(MQ6);
temp7I[ind] = analogRead(MQ7I);
temp7II[ind] = analogRead(MQ7II);
temp9[ind] = analogRead(MQ9);
if(ind>1)
{
  if(flag2==0)
  x2 = temp2[ind]-temp2[ind-1];
  x4 = temp4[ind]-temp4[ind-1];
  x6 = temp6[ind]-temp6[ind-1];
  x7I = temp7I[ind]-temp7I[ind-1];
  x7II = temp7II[ind]-temp7II[ind-1];
  x9 = temp9[ind]-temp9[ind-1];
  flag2 = 1;
}

if(flag2==1)
{
  y2 = temp2[ind]-temp2[ind-1];
  y4 = temp4[ind]-temp4[ind-1];
  y6 = temp6[ind]-temp6[ind-1];
  y7I = temp7I[ind]-temp7I[ind-1];
  y7II = temp7II[ind]-temp7II[ind-1];
  y9 = temp9[ind]-temp9[ind-1];
  flag2 = 0;
}


if(ind >2)
{
  Serial.print("x2 = ");
  Serial.println(x2);
  Serial.print("y2 = ");
  Serial.println(y2);
  Serial.print("x4 = ");
  Serial.println(x4);
  Serial.print("y4 = ");
  Serial.println(y4);
  Serial.print("x6 = ");
  Serial.println(x6);
  Serial.print("y6 = ");
  Serial.println(y6);
  Serial.print("x7I = ");
  Serial.println(x7I);
  Serial.print("y7I = ");
  Serial.println(y7I);
  Serial.print("x7II= ");
  Serial.println(x7II);
  Serial.print("y7II = ");
  Serial.println(y7II);
```

```
      Serial.print("x9 = ");
      Serial.println(x9);
      Serial.print("y9 = ");
      Serial.println(y9);
      Serial.println("");
      if(abs(x6-y6)<4)
      //if(abs(x2-y2)<2 && abs(x4-y4)<2 abs(x6-y6)<2
      {
         flag = 1;
         digitalWrite(yLEDpin,LOW);
      }
    }
    delay(20000);
    ind++;
  }
  Serial.println("Out of Loop!");
  flag = 0;
}

void loop()
{

  File datafile = SD.open("datafile.txt",FILE_WRITE);
  //declares an object of the File class

  if(SD.exists("datafile.txt"))     //checks if file exists on card
  {
    digitalWrite(gLEDpin,HIGH);
    //lights green LED and turns red LED off if file exists
    digitalWrite(rLEDpin,LOW);
    Serial.println("File Found");
    SD.open("datafile.txt");        //opens the file for writing
    if (flag == 0)                  //runs first time, then never again
    {
      SD.open("datafile.txt");
      datafile.println("time,MQ2,MQ4,MQ6,MQ7I,MQ7II,MQ9");
      datafile.close();
      flag = 1;
    }
    //opens file for writing
    //records column headers
    //closes file for writing
    //changes flag to record data

    if(flag == 1)  //runs through this part of the loop every time after
    {              //the first time, where it writes the column headers
      Serial.println("File Opened");
      datafile.print(time);
      datafile.print(",");
      datafile.print(analogRead(MQ2));
      datafile.print(",");
      datafile.print(analogRead(MQ4));
      datafile.print(",");
      datafile.print(analogRead(MQ6));
      datafile.print(",");
```

```
      datafile.print(analogRead(MQ7I));
      datafile.print(",");
      datafile.print(analogRead(MQ7II));
      datafile.print(",");
      datafile.println(analogRead(MQ9));
      Serial.println("File Written");
      datafile.close();                    //closes the file for writing
      Serial.println("File Closed");
      //records the total time elapsed (seconds)
      //delimits the data with commas for exporting
      //records the sensor values for each sensor
    }
  }

  else
  {
    digitalWrite(gLEDpin,LOW); //lights red LED if file doesn't exist on SD card
    digitalWrite(rLEDpin,HIGH);
    Serial.println("404: File Not Found");
  }

  Serial.println(analogRead(MQ2));
  Serial.println(analogRead(MQ4));
  Serial.println(analogRead(MQ6));
  Serial.println(analogRead(MQ7I));
  Serial.println(analogRead(MQ7II));
  Serial.println(analogRead(MQ9));

  delay(inc);              //delays for however many seconds inc is set as
  time += abs(millis()); //increments time
  Serial.println("");      //changes to a new line for next data entry
  while(time>8000)
  {
    digitalWrite(gLEDpin,LOW);
    delay(1000);
    digitalWrite(gLEDpin,HIGH);
    delay(1000);
  }
}
```

Appendix B: MATLAB Scripts and Functions

The MATLAB script that was used to produce the many of the graphs is shown below.

```
    commentstyle
m6 = MQ6';        %Assigns the transpose of the MQ6 vector to m6
m7I = MQ7I';      %Assigns the transpose of the MQ7I vector to m7I
m7II = MQ7II';    %Assigns the transpose of the MQ7II vector to m7II

p2 = polyFit(time,m6,1);          %Finds the line of best fit for the m6 vector
p7I = polyFit(time,m7I,1);        %Finds the line of best fit for the m7I vector
p7II = polyFit(time,m7II,1);      %Finds the line of best fit for the m7II vector

[Time,y6] = SXY(p6,min(time),(max(time)-min(time))/length(time),max(time)-1);
[Time,y7I] = SXY(p7I,min(time),(max(time)-min(time))/length(time),max(time)-1);
[Time,y7II] = SXY(p7II,min(time),(max(time)-min(time))/length(time),max(time)-1);
%Creates vectors to plot line of best fit for each signal

y6 = y6';         %Transposes y6
y7I = y7I';       %Transposes y7I
y7II = y7II';     %Transposes y7II

plot(time,MQ2,time,MQ4,time,MQ6,time,MQ7I,
        time,MQ7II,time,MQ9,Time,y2,Time,y7I,Time,y7II)
legend('MQ2','MQ4','MQ6','MQ7I','MQ7II','MQ9',
        'MQ2 Trend','MQ7I Trend','MQ7II Trend')
%Plots data and displays a legend
xlabel('Time (ms)')                       %Labels x-axis
ylabel('Sensor value (0 - 1023)')         %Labels y-axis
title('Propane 2:20 Mixture - 100k')      %Creates plot title
grid on                                   %Turns on gridlines
```

Since it is not a function built into MATLAB, the S to X,Y function used in the SD card Arduino code (SXY) is shown below.

```
    commentstyle
function [ x,y ] = SXY( S,Min,Inc,Max )
%
% Given the solution vector S from the polyFit function (a
% vector containing the coefficients of a polynomial function),
% this function returns two vectors representing the inputs and
% outputs of that function, which can be plotted.
%
% Variables:
%       x: returned independent variable
%       y: returned dependent variable
%       S: solution vector from the polyFit function
%       Min: the lower bound for the independent variable
%       Inc: the increment for the independent variable
%       Max: the upper bound for the independent variable
%       n: the length of S (the degree of polynomial + 1)

[x] = Min:Inc:Max;          % Creates x based on Min, Inc, and Max
[y] = x.*0;                 % Creates a zero vector y the same length as x
n = length(S);              % Creates n from the length of S
for l = 1:n                 % Additively creates y with each iteration being
        y = y + S(l).*x.^(l-1); % the next level of polynomial
        l = l + 1;          % increments to the next level of polynomial
end
end
```

Also, it should be noted that the polyFit function mentioned is not the built in polyfit function in MATLAB, but one that was created as practice when learning to use the program. While both are equally effective, the built in function produces a 1xn vector, while the one used here produces an nx1 vector, so the transpose of that solution would be needed in order to use it as an argument for this function.

The following function was used to calculate β in the equations described previously and produces an array of values that can be plotted to view the best-fit curve.

commentstyle

```
function [ mqExpFit ] = resFit( conc,res,R0,Rinf,mqx )
% resFit is the function that calculates the best-fit equation coefficients
% for a given array of sensor values. Given the different concentrations it
% is tested at (conc), the values of the sensitivity resistors (res), the
% values of R_0 (R0) and R_{\infty} (Rinf) for each resistor setting, and
% the observed sensor readings (mqx), it returns an array of values for the
% best-fit curve at the same concentrations at which the sensor was tested
% (mqExpFit).

Vmax = 5;                   % declares the maximum voltage
Smax = 1023;                % declares the maximum sensor signal
I = zeros(10);              % creates an array for the current
Rv = zeros(10);             % creates array for the res. across the sensor
mqV = mqx*5/1023;

for i = 1:10                            % for each sensitivity setting
    for j = 1:10                        % for each concentration
        Vv(i,j) = Vmax - mqV(i,j);  % calculates voltage drop across sensor
        I(i,j) = mqV(i,j)/res(i);   % calculates current in circuit
        Rv(i,j) = Vv(i,j)/I(i,j);   % calculates resistance across sensor
    end
end

sums = conc*0;              % creates an array of zeros the same size as conc
for i = 1:10
    for j = 1:10
        sums(i) = sums(i)+log((R0(i)-Rinf(i))/(Rv(i,j)-Rinf(i)))/(conc(j));
    end         % calculates the sum term used to calculates $\beta$
end

beta = .1*sums;             % calculates $\beta$

mqExpFit = zeros(10);   % creates an array of zeros to hold the best-fit values
for i = 1:10
  for j = 1:10
    mqExpFit(i,j)=Smax*res(i)
    /(res(i)+Rinf(i)+(R0(i)-Rinf(i))*exp(-beta(i)*conc(j)));
    % uses best-fit equation to calculate the best-fit values
  end
end


x = 0:1:1023;
for i = 1:10
    for j = 1:10
         y(i,j)=-1/beta(i)*log((Smax*res(i)-mqx(i,j)*res(i)-mqx(i,j)
         *Rinf(i))/(mqx(i,j)*(R0(i)-Rinf(i))));
    end         % creates data for transformed equation to get concentration as
end             % a function of sensor reading (to check for accuracy in fit)

for i = 1:10    % concurrently plots the observed data, the best-fit curve,
```

```
    plot(mqx(i,1:10),conc,'.')  % and a plot generated from the transformed
    grid on     % equation which is a function of sensor reading, not
    hold on % concentration
    pause
    plot(mqExpFit(i,1:10),conc)
    pause
    plot(mqx(i,1:10),y(i,1:10))
    pause
end

end
```

University Honors Program

Capstone Approval Page

Capstone Title (print or type)
Characterization of MQ-Series Gas Sensor Behavior

Student Name (print or type) Alec Fisher

Faculty Supervisor (print or type) Martin Kocanda

Faculty Approval Signature Martin Kocanda / per phone approval

Department of (print or type) Electrical Engineering

Date of Approval (print or type) 5/10/13