Gaseously Stressful Times

Prepared for: Owen Mundy, Professor

Prepared by: Jack Clary, Student

May 11, 2018

ABSTRACT SUMMARY

With the hopes of determining stressful times of day on campus, I used a Raspberry Pi, MQ-4/MQ-8 sensors to track gas levels in the air. The hope was to correlate elevated gas levels on campus with elevated levels of stress. This data was going to be corroborated by student data gathered by means of notes placed in the housing unit of the Pi.

EXECUTIVE SUMMARY

Objective

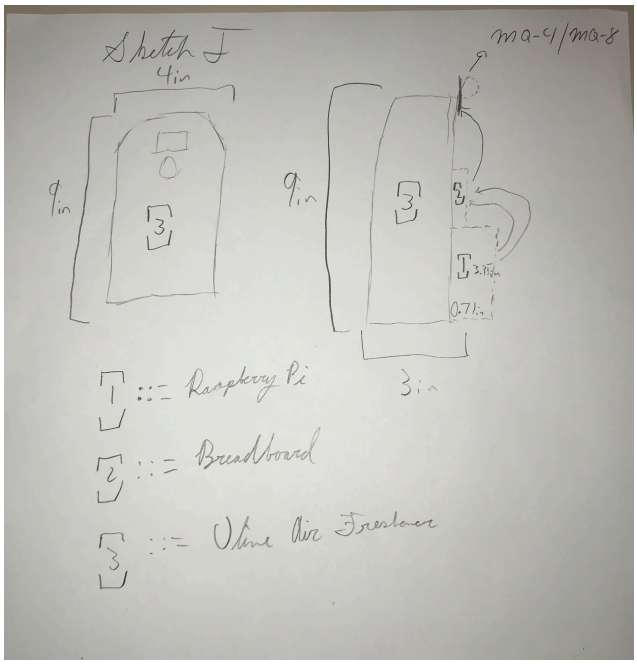
How could one measure the most stressful time of the day in a non-subjective quantitative sense? Researchers could attempt to request information from individuals across a specific population, asking something similar to "How are you feeling today?", but with that methodology comes a level of subjectively that does not satisfy the criteria I have laid forth. Therefore, how could someone measure the stress in an area? My hypothesis is predicated on this fact: as humans become stressed, their gastrointestinal tract becomes agitated, thus forcing individuals firstly, to use the restroom more, and secondly to release more gaseous material into the air.
With that being said, my hypothesis is this: the times at which specific gas levels are highest in an enclosure will correlate with the general level of stress in the given enclosure.

Goals

1. Create a device that measures CO, CH4, LPG, and H2.
2. Disguise the device so that it is not out of place in its environment.
3. Create a program to gather the data of the amount of the four chemicals in the air, write them to a csv, email the csv to my Davidson College email address, and visualize the data in a time-series graph.

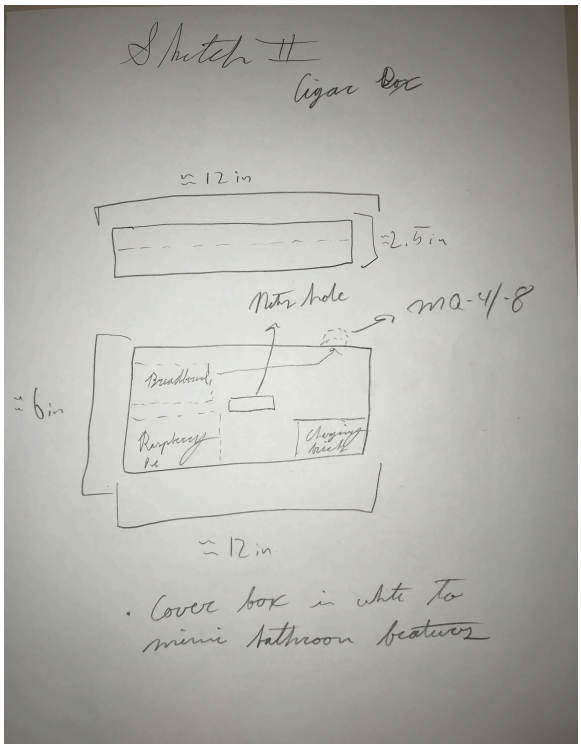
Process

Original Sketches and Design



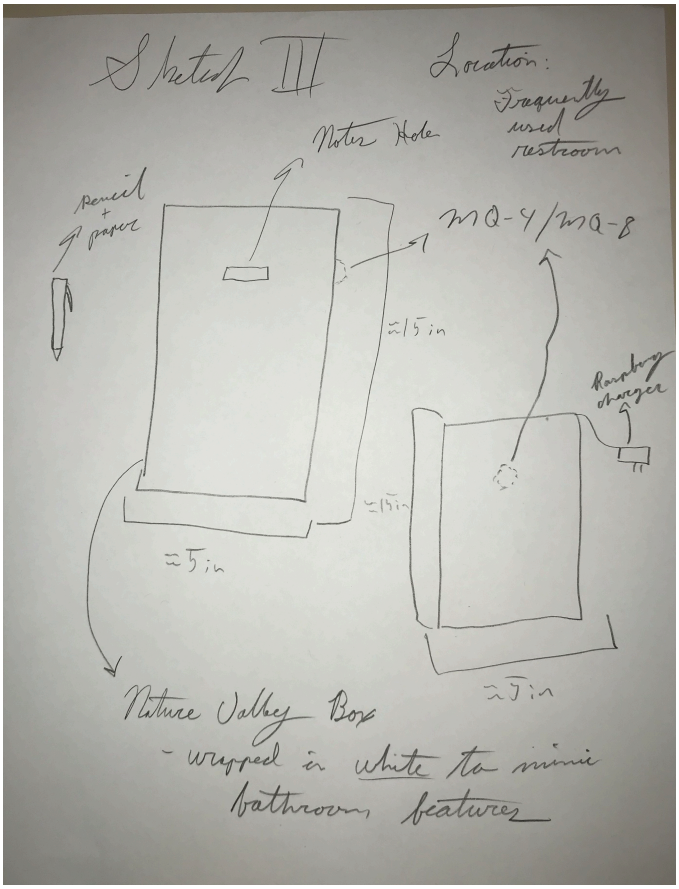
Sketch I:

The original design for the sensor was to attach it to the back of a Uline Air Freshener, an object typically found in the restrooms of schools and other busy enterprises. This design fell from grace due to the design not allowing for user submitted data to corroborate the data the sensor collected.



Sketch II:

The second design was comprised of a cigar box with a hole cut in the rear and in the lid. The first for the gas sensor to protrude, and the second for users to submit notes for data gathering. This design was well underway until the lid of the box cracked upon sawing into it. Just as with the first design, this box would have been covered in a white material, but unlike the air freshener, would have housed a battery for the Pi.



Sketch III and Final Design Choice:

The final design is comprised of a 15 inch tall box covered in white to mimic the surrounding area, i.e. a bathroom's tiles and walls, so as to be as inconspicuous as possible. It has a hole in the front side for passing in notes, a hole on the right side to expose the MQ sensor, and a small hole in the top to feed the Raspberry Pi's power cable through.

Overall Design

The overall design, i.e. choice of sensor(s), location, and disguise are as follows: the MQ-4 and MQ-8 sensors sense primarily CO, CH4, LPG, and H2 allowing prime gas levels released to be monitored, the location of the bathroom was chosen because with increased stress comes increased bathroom use, and the disguise was chosen to mimic the color scheme of the restroom while adding the ability to gain data from voluntary participants.

Project Outline

Device: Raspberry Pi V.3 and MQ-8 Sensor (Originally a MQ-4 sensor)

Location: Male restroom that is frequently used

Camouflage: A General wellness note collection box

-A box that students place pieces of paper in that has the time and their current stress level

-Stressed or Not Stressed

Visualization: Time series graph to easily show spikes in gas levels. (x-axis: gas levels, y-axis: time of day)

Project Process Summary

1. Aquire the Raspberry Pi, MQ-X Sensor, MCP3008, and female to male headers.
    1. Find whatever MQ-X sensor you are using's data sheet online (google "MQ-X datasheet")
2. Flash the Raspberry Pi using Etcher
    1. Download Etcher https://etcher.io
    2. Download Raspbian Disk Image https://www.raspberrypi.org/downloads/raspbian/
    3. Plug in a SD card to your computer
    4. Run Etcher
    5. Drag the disk image into the Etcher window and choose the correct SD card from the device list.
    6. Flash the disk image onto the Raspberry Pi
3. Insert the SD card into the Raspberry Pi
4. Follow these instructions to enable SSH on the Pi https://www.raspberrypi.org/documentation/remote-access/ssh/
5. Go to raspi-config (command line: sudo raspi-config)
    1. Go the the Interfacing menu and enable SPI
    2. Reboot the Pi (command line: sudo reboot)
6. Install python3 dev on the Pi (command line: sudo apt-get install python3-dev)
7. Install master.zip for spidev (command line: wget https://github.com/Gadgetoid/py-spidev/archive/master.zip)

1. Unzip master.zip (command line: unzip master.zip)
2. Remove the zip file (command line: rm master.zip)
3. cd into the unzipped file
4. Run this command to install spidev (command line: sudo python3 setup.py install)

8. cd to whatever directory you want to code in
9. Git clone via the command line these files into your Raspberry Pi: git clone https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-MQ
    1. You will have to update the MQ.py file to support whatever MQ sensor you are using. The documentation in the file will point you in the correct direction using your MQ-X datasheet
10. Follow the connection rules in Figure 1 in order to connect the MCP3008 and the MQ-X to Raspberry Pi.
11. ssh into your Raspberry Pi, navigate to the git cloned file Raspberry-Pi-Gas-Sesnor-MQ and run example.py (command line: python3 example.py)
12. If the code operates correctly and you wish to have the code run while you are not connected follow the below steps
13. Install tmux on your Raspberry. (command line: sudo apt-get install tmux)
14. cd into Raspberry-Pi-Gas-Sesnor-MQ
15. Run tmux (command line: tmux)
16. Run your code from step 11
17. Exit your command line shell
18. To reconnect to your tmux session (command line: tmux att)
    1. A list of tmux commands can be found here: https://gist.github.com/henrik/1967800

Writing to a CSV

1. In example.py add this at the beginning of the file: import csv
2. Then implement this code where you see fit. I've added the datetime module as well as the csv module.
    1. My entire code is available at https://github.com/jaclary/DIG333Final/tree/master/Scripts

```
    with open('data.csv', 'a', newline='') as csvfile:
        csvData = csv.writer(csvfile, delimiter=' ',
                quotechar='|', quoting=csv.QUOTE_MINIMAL)
        dateString = datetime.datetime.strftime(datetime.datetime.now(),"%Y-%m-%d %H:%M:%S")
        csvData.writerow([dateString, float(perc["GAS_LPG"]), float(perc["CO"])])
```

Sending email from the Raspberry Pi

1. Install mailutils and ssmtp
    1. sudo apt-get install ssmtp
    2. sudo apt-get install mailutils
2. Navigate and edit the ssmtp config file (command line: sudo nano /etc/ssmtp/ssmtp.conf)
3. (NO QUOTES) Add “AuthUser=YourEmail”
4. (NO QUOTES) Add “AuthPass=YourPassword”
5. (NO QUOTES) Add “smtp.gmail:587” to “mailhub=“
6. (NO QUOTES) Add “UseSTARTTLS=YES”
7. Add code to example.py:

```
import smtplib, mimetypes
from email.mime.multipart import MIMEMultipart
from email import encoders
from email.message import Message
from email.mime.audio import MIMEAudio
from email.mime.base import MIMEBase
from email.mime.image import MIMEImage
from email.mime.text import MIMEText

emailfrom = "sender@example.com"
emailto = "destination@example.com"
fileToSend = "hi.csv"
username = "user"
password = "password"

msg = MIMEMultipart()
msg["From"] = emailfrom
msg["To"] = emailto
msg["Subject"] = "help I cannot send an attachment to save my life"
msg.preamble = "help I cannot send an attachment to save my life"

ctype, encoding = mimetypes.guess_type(fileToSend)
if ctype is None or encoding is not None:
    ctype = "application/octet-stream"

maintype, subtype = ctype.split("/", 1)

if maintype == "text":
    fp = open(fileToSend)
    # Note: we should handle calculating the charset
    attachment = MIMEText(fp.read(), _subtype=subtype)
    fp.close()
elif maintype == "image":
    fp = open(fileToSend, "rb")
    attachment = MIMEImage(fp.read(), _subtype=subtype)
    fp.close()
elif maintype == "audio":
    fp = open(fileToSend, "rb")
    attachment = MIMEAudio(fp.read(), _subtype=subtype)
    fp.close()
else:
    fp = open(fileToSend, "rb")
    attachment = MIMEBase(maintype, subtype)
    attachment.set_payload(fp.read())
    fp.close()
    encoders.encode_base64(attachment)
attachment.add_header("Content-Disposition", "attachment", filename=fileToSend)
msg.attach(attachment)
```

```
server = smtplib.SMTP("smtp.gmail.com:587")
server.starttls()
server.login(username,password)
server.sendmail(emailfrom, emailto, msg.as_string())
server.quit()
```

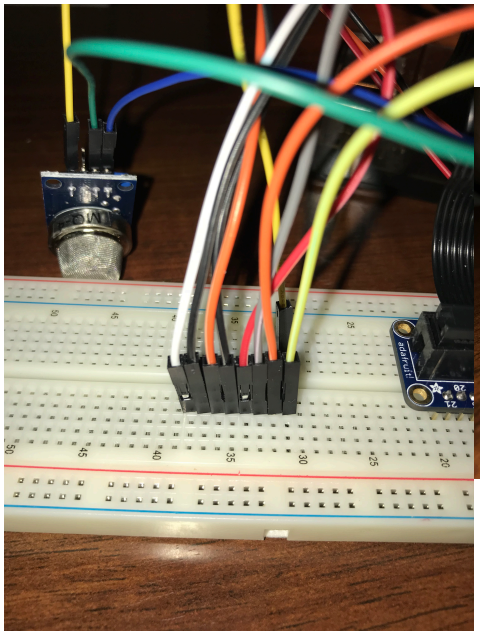
Code Source: https://stackoverflow.com/questions/23171140/how-do-i-send-an-email-with-a-csv-attachment-using-python

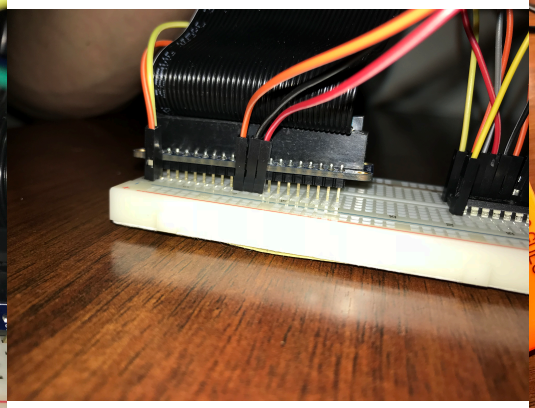
Device

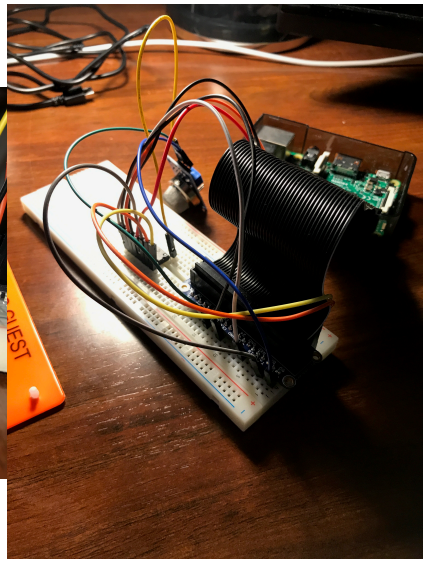
The devices used are a Raspberry Pi, a breadboard, multiple female to male wires, a MCP3008 ADC converter, and a MQ-8 gas sensor.

Schematic/Photos

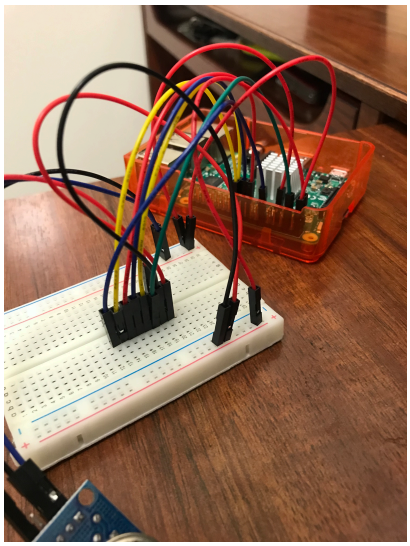
Original

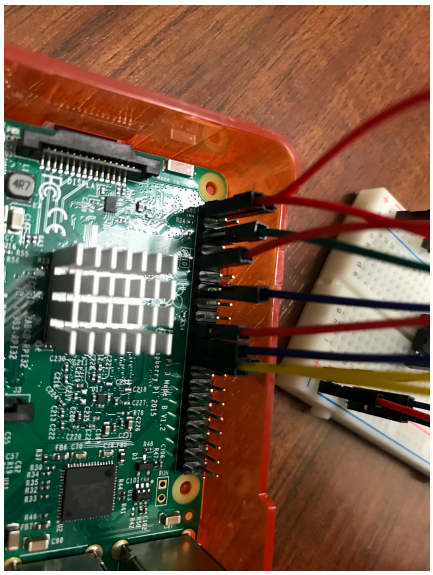


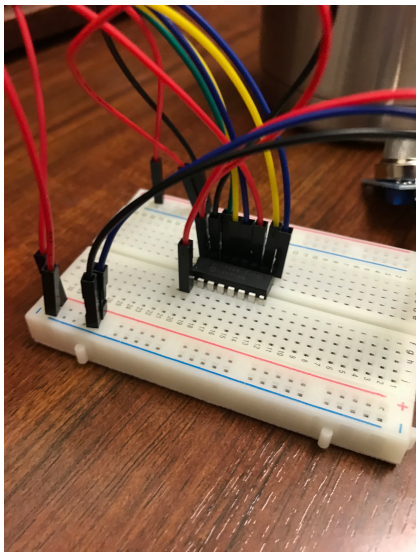




Final







Schematic Explanation (Figure 1)

MQ-8	Raspberry Pi V.3
VCC	5V
GND	Any GND on the Pi
AOUT	CH0 (MCP3008)

MCP3008	Raspberry Pi V.3
VDD	3.3V
VREF	3.3V
AGND	GND
DGND	GND
CLK	Pin 18

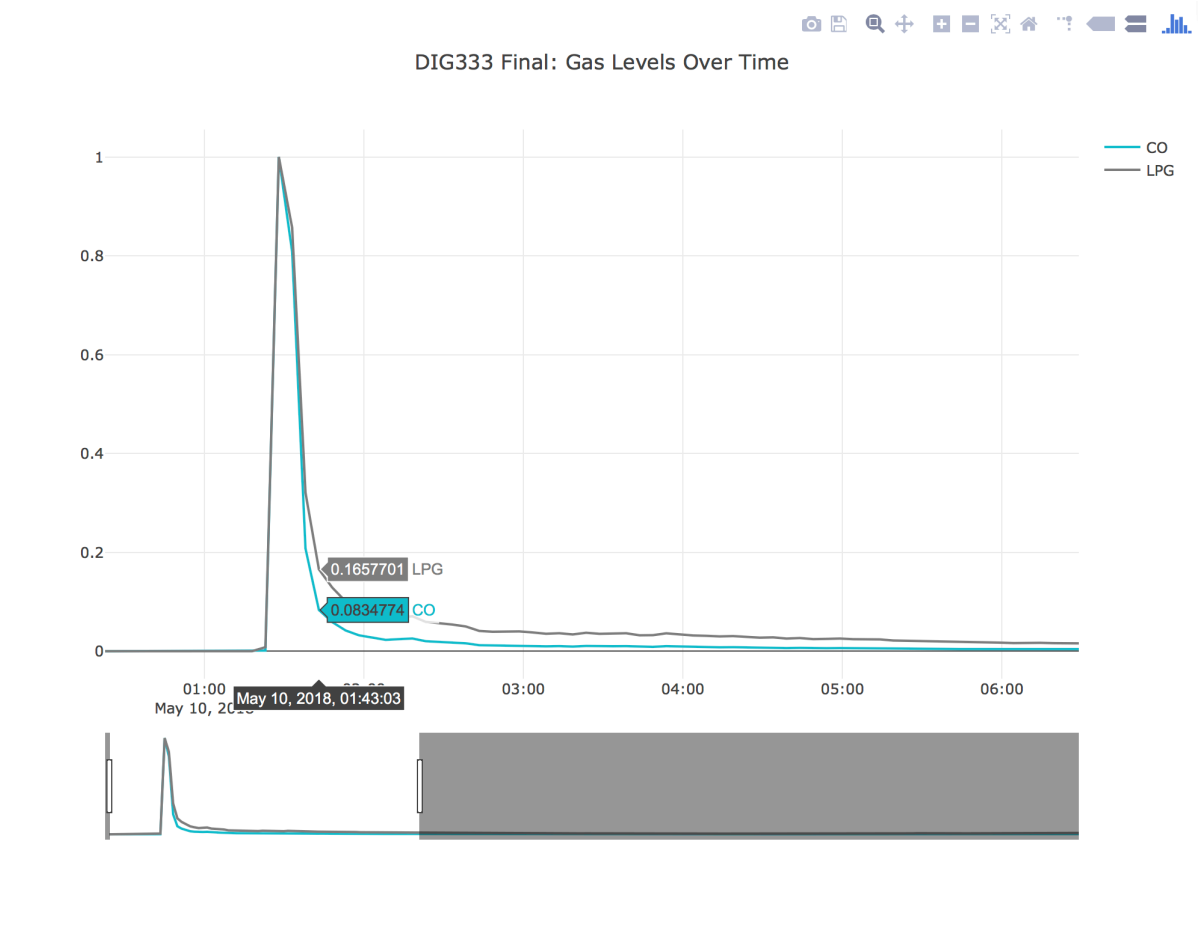
MCP3008	Raspberry Pi V.3
DOUT	Pin 23
DIN	Pin 24
CS/SHDN	Pin 25

Internal Program(s): All programs are written in Python 3

- MCP3008.py
  - This is a class that was derived from a similar project by Felix on tutorials-raspberrypi.de
  - It allows for reading in data from the MCP3008 ADC converter.
- MQ.py
  - Class that handles calculating the sensor's resistance, calibrating the sensor, reads in the gas sensor's data and calculates the specific gas values in ppm.
  - The respective calculations are derived from the sensors data sheet. (Technical Data MQ-8 Gas Sensor)
- SpiDev Module
  - spidev is a module that allows interfacing between serial port interfaces that occur on the Raspberry Pi.
- example.py
  - This is the main function for the program. It calls the MQ class thus allowing data capture from the MQ-8 sensor.
  - Functions
    - Aquire Data
    - Write Data to a CSV - "data.csv"
    - Email "data.csv" to jaclary@davidson.edu

External Programs: Data Visualization

- Time Series Graph showing LPG and CO levels over time
  - This can easily be adjusted to accommodate more data
- Highlights
  - Interactive graph that allows zoom, mouse over overlays, data line selection, and downloadability.
- Programing Languages used: HTML, CSS, JavaScript(D3 and plot.ly)
- Implementation
  - Import plot.ly and d3
  - Augment data from CSV through functions (unpack(), unpackInitial(), and numberUnPack())
    - Normalize CO and LPG data from [0-1]

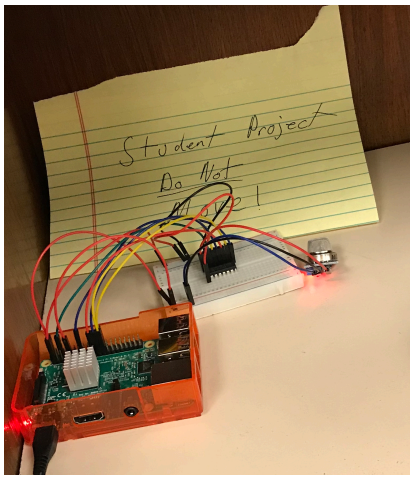


- Formula : where z is the normalized value : $z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$
- Combine Date and Time fields into one object

- Assign data to the x-axis and y-axis by creating a variable named trace1 and trace2
  - Trace1
    - x : CO level
    - y : Date and Time
  - Trace2
    - x : LPG level
    - y : Date and Time
- Plot them using Plotly.newplot("where to draw", "what data to use", "what layout to use")

Final Conclusion and Outcome

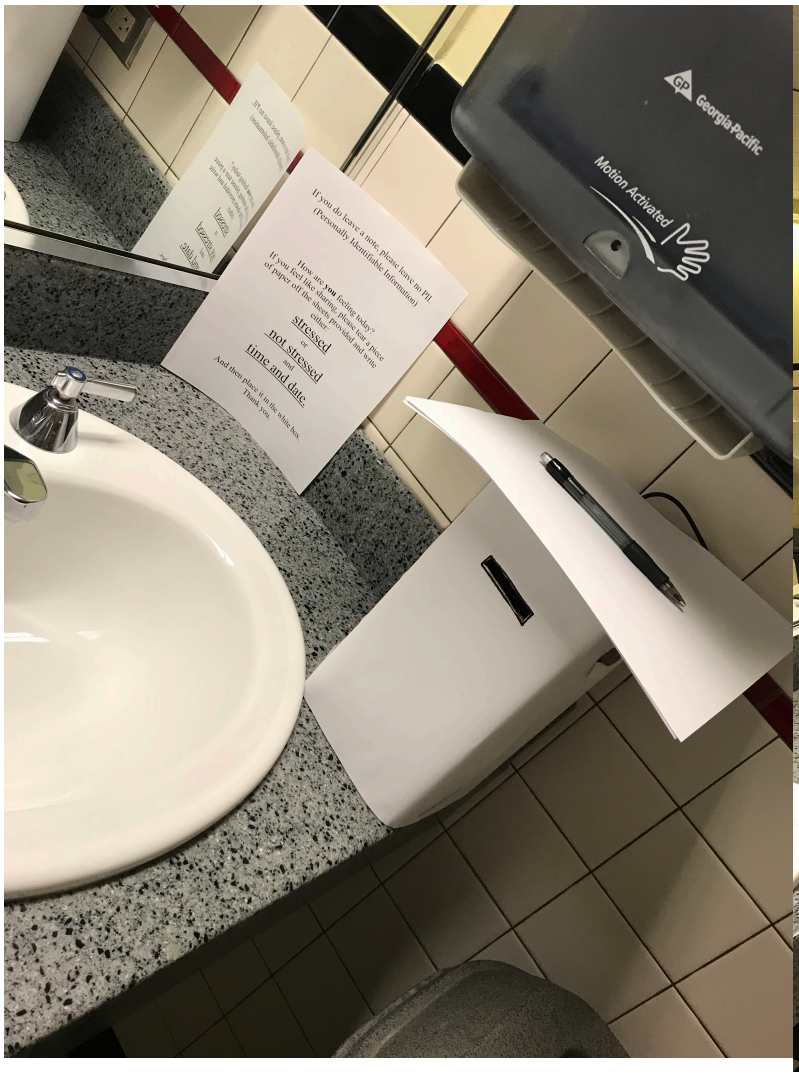
Due to an unknown individual stealing the final design as shown in the photo on the first page, and due to lacking another container and being short on time, the final product used only an MQ-4 sensor and was hidden in plain sight as shown in the photo below.



Overall the project, despite being started from scratch with little time remaining due to the disappearance of the original device, turned out relatively well. The second device did everything the first device, including capturing data, writing it to a csv, and emailing to myself, but not including sensing the specific gases the original device did and being housed in a box designed to disguise it in its proper location.

For reference, all sources and files can be found at this GitHub address: https://github.com/jaclary/DIG333Final

On the following page are photos of the original device in its intended habitat.


Georgia-Pacific
Motion Activated
If you do leave a note, please leave no PII
(Personally Identifiable Information)
How are you feeling today?
If you feel like sharing, please tear a piece
of paper off the sheets provided and write
either
stressed
or
not stressed
and
time and date.
And then place it in the white box.
Thank you.