- Team contract: EID101B-Team-Contract- Team 4.pdf

Sep 30 (Incredible Beginning)

- We discussed our problem statements and decided to focus on the issue of food waste in urban areas such as New York City.

Oct 7

- We split roles for our background research draft and created a shared document for compiling sources. We also worked on our ethics assignment.
    - Background research sources: BACKGROUND SOURCES
    - Background research draft: Team 4 Background Research Doc

Oct 14

- We worked on our first Gantt chart draft by referencing deadlines on the class syllabus.
    - Gantt chart: Online Gantt 20241014.pdf
- We also divided groups to work on our three prototype drafts: Zeshui and Chris, Zehra and Somin, Allen and Ahikara.

Oct 17

- Reviewed our background research draft and added correct citations and bibliography before class started

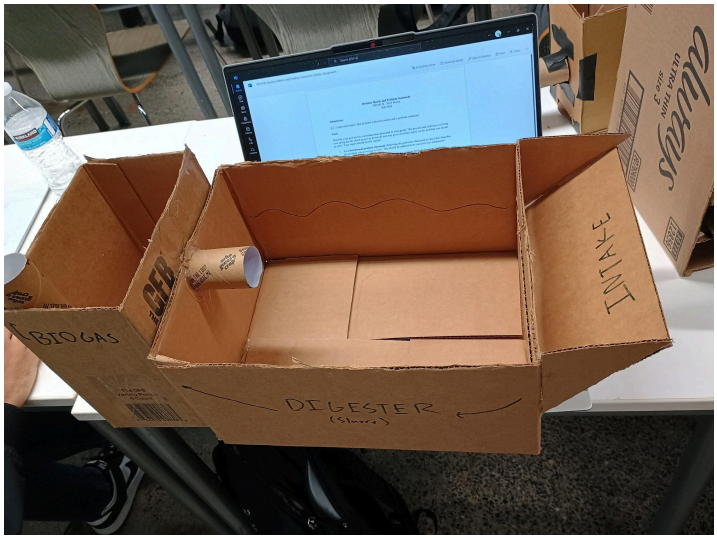
Oct 21 (Ramping up into our Peak)

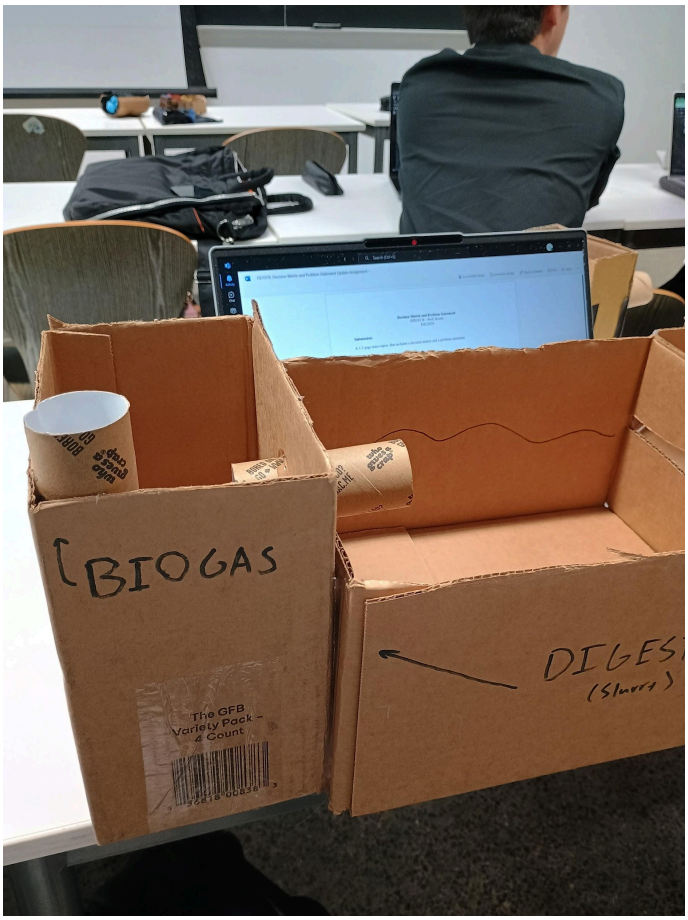
- Reviewed prototype ideas and worked in the maker space to build and design them
    - The layout and feasibility of a biogas system (Prototype 1)
    - A dehydrator and grinder in the process of eco-brick formation (Prototype 2)
    - Contraption containing sensors to model freshness indicators that detect how if food is close to going bad (Prototype 3)

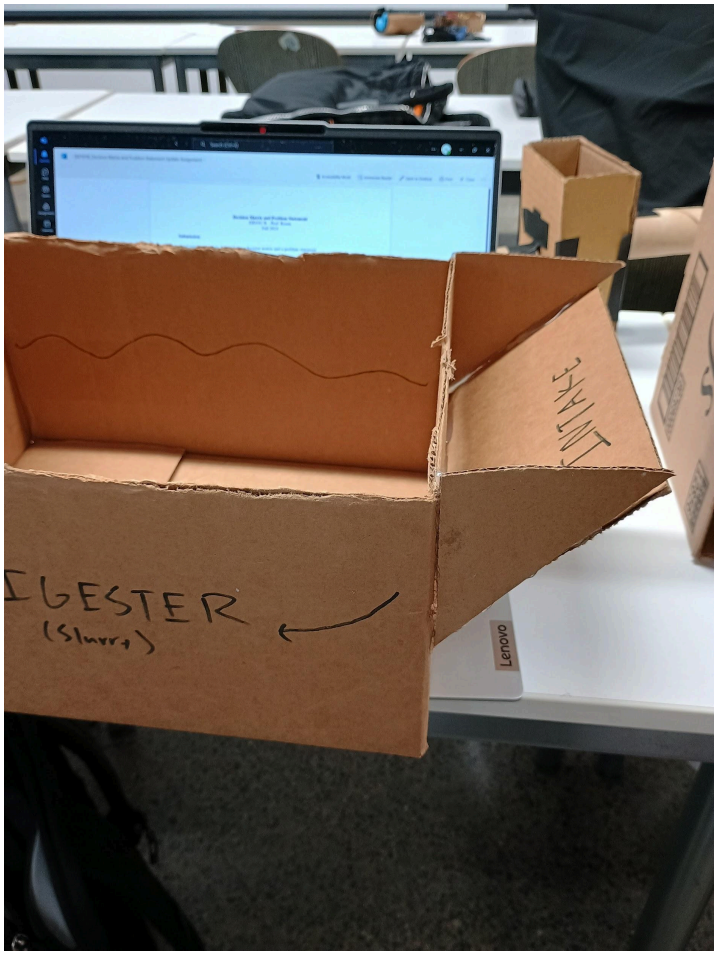
- Created a “bio-brick” prototype (dehydrator, blender, compression system)
(put images later)

Oct 24 (Prototyping Madness)

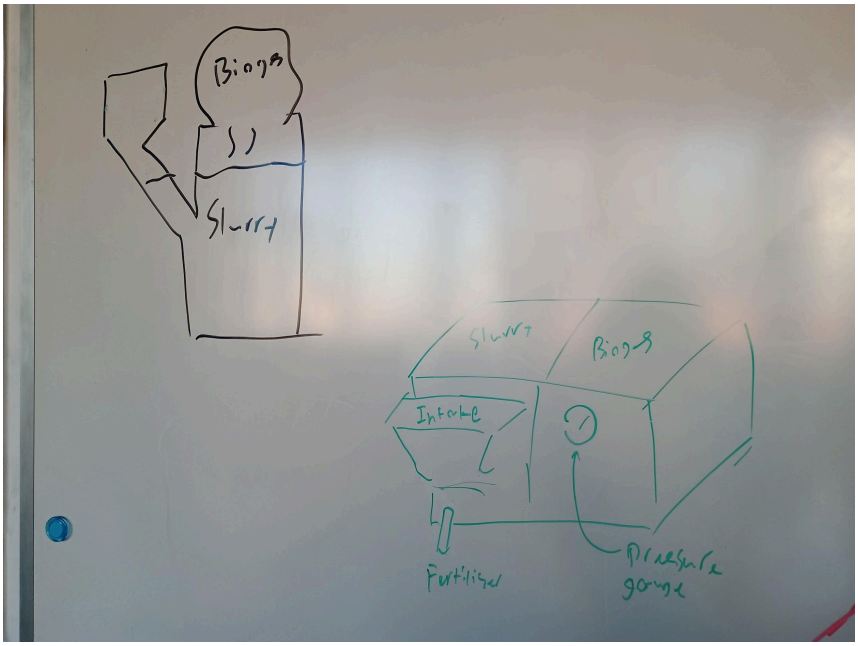
- **Prototype 1 BIOGAS**



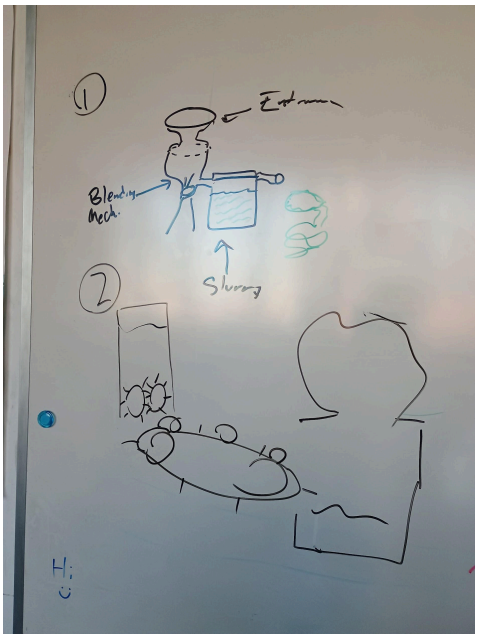


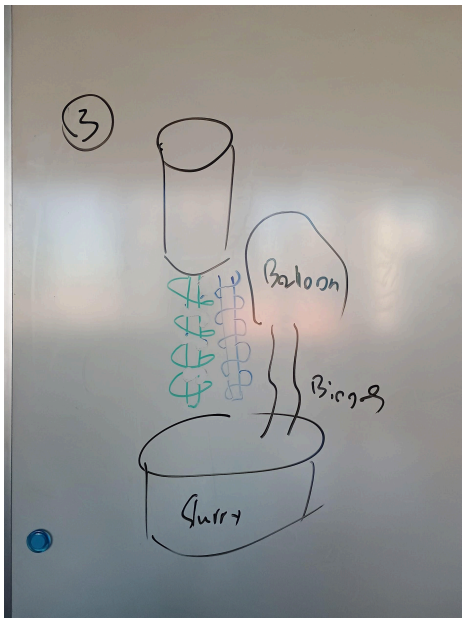


    - Final Design sketch



- Our idea was to have the intake setup for liquid food waste. This way, the waste can flow into the digester without needing extra processing. The waste will be fed into the slurry solution, where anaerobic respiration and biogas will be produced. We will have a spout so the leftover slurry can be used as fertilizer, and the biogas chamber will have a pressure gauge for safety purposes.





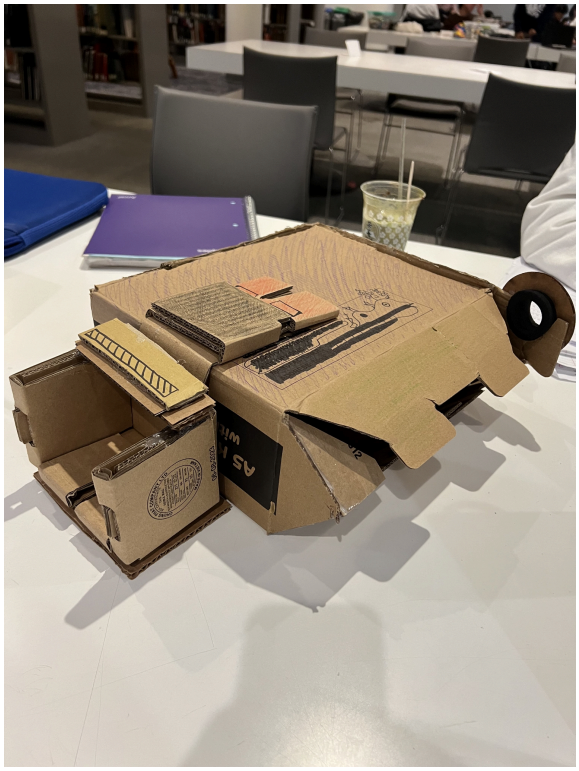
- Draft sketches
    - Sketch 1

        - This shows a basic layout for a biogas reactor design. A blender will be in the food waste container, liquifying the waste and feeding it into a slurry.
    - Sketch 2
        - This video shows another method of transferring food waste and liquifying it with interlocking gears instead of a high-speed blender. This should help improve safety.
    - Sketch 3
        - This is a modification to sketch 2, where the transferring and liquefaction of waste are combined into a corkscrew design with two corkscrews going in opposing directions.

- **Prototype 2 ECO-BRICKS**

  - Top left - prototype of the shredder/blender to blend dehydrated food waste into powder
  - Top right: Dehydrator; removes moisture from food, leaving behind dry flesh. It has trays lined vertically for easy separation of food waste and removal.
  - Bottom right - the shredded food waste (now in powder form) in a mold in the shape of a brick

- **Prototype 3 FRESHNESS INDICATOR**



Oct 28 (Calm before the storm)

- Completing the decision and problem statement update
  Decision Matrix and Problem Statement Update
- Group meeting to work on midterm presentation slides

Nov 4

- Decide what sensor to use (CO2, NH3, H2S?)
  - Decided to go with CO2?
  - Researched different CO2 sensors'
- DIY CO2 Sensor Guides
  - https://atlas-scientific.com/blog/co2-meter-arduino/
  - https://simplyexplained.com/blog/i-built-a-co2-sensor-and-it-terrifies-me/
  - https://www.instructables.com/DIY-Carbon-Dioxide-CO2-Detector/

Project Plan for a Compact Freshness Indicator

1. Components Needed

Component	Approximate Cost
CO_2 Sensor (NDIR)	$50–$150
Microcontroller	$25–$40
OLED or LED Display	$10–$20
Battery and Charger	$10–$30
Storage (SD Card Module)	$10
Casing (3D Printed or DIY)	$5–$15
Breadboard & Wires	$5–$10

Total estimated cost: **$110–$275**

2. Selecting and Setting Up the CO_2 Sensor

- **Sensor Choice:** Non-dispersive infrared (NDIR) CO_2 sensor provides reliable readings over time and works well for measuring CO_2 buildup. MTP60-A Infrared NDIR Carbon Dioxide Sensor intelligent small sensor - Walmart.com

- **Programming:** Connect the sensor to an Arduino or Raspberry Pi. Many NDIR sensors have libraries for easy integration, allowing you to log data and perform fundamental analysis.
- **Sensitivity:** Look for a sensor with a detection range of around 400–5000 ppm (parts per million), which is typical for air quality sensors and suitable for tracking CO_2 changes during spoilage.

3. Building the Device

- **Hardware Assembly:**
    - Connect the CO_2 sensor to the microcontroller.
    - Add a display (like a small OLED) to show real-time CO_2 readings and spoilage alerts.
    - Use a battery for portability, with a rechargeable setup if possible.
    - Include an SD card module to log CO_2 data over time if your microcontroller does not have enough storage.
- **Casing:** Use a compact case to house all components. A small vented compartment for the sensor allows airflow while protecting the rest of the electronics. A DIY plastic casing or a 3D-printed design would work well.

4. Software & Data Logging

- **Data Collection:** Write a program on the Arduino or Raspberry Pi to log CO_2 levels continuously continuously. Set it to record data at intervals (e.g., every 10 minutes) to capture the gradual buildup of CO_2.
- **Calibration and Baseline:** Record an initial CO_2 baseline for each vegetable upon purchase, then monitor it over several days to observe the change as the vegetable begins to spoil.
- **Determining the Ratio Threshold:** Calculate the ratio of the current CO_2 level to the initial baseline level. Through repeated measurements with your 5 sample vegetables, identify the approximate CO_2 ratio that correlates with spoilage.
- **Display and Alert:** Program the device to display the real-time CO_2 reading. When the ratio reaches a critical threshold (based on your data), have the display show an alert (e.g., “Spoiling”) to indicate that the vegetable may be going bad.

5. Testing and Data Collection

- **Vegetable Selection:** Choose five vegetables with different spoilage rates, such as tomatoes, cucumbers, bell peppers, carrots, and spinach. These can provide a good range of spoilage behavior and gas emission rates.
- **Storage Conditions:** Store each vegetable sample in a small, sealed container with the CO_2 sensor placed near the vegetables to track CO_2 emissions directly. Ensure the

containers are ventilated slightly or periodically to mimic typical fridge or counter storage conditions.

- **Data Analysis:** Use the collected data to analyze the CO_2 trends. For each vegetable, note when spoilage becomes noticeable (e.g., soft spots, color change) and correlate this with the recorded CO_2 ratio.

6. Example Code Structure (Arduino Pseudo-code)

Here's a sample structure for the Arduino code:

```
#include <CO2SensorLibrary.h>  // Hypothetical library for the CO₂ sensor
#include <SD.h>
#include <OLED.h>              // OLED display library

#define CO2_SENSOR_PIN A0
#define RATIO_THRESHOLD 1.5   // Placeholder threshold; to be adjusted after data collection
#define BASELINE_INTERVAL 10  // Time in minutes to log baseline readings

float initialCO2 = 0;
float currentCO2 = 0;
float ratio = 0;

void setup() {
    Serial.begin(9600);
    CO2Sensor.begin(CO2_SENSOR_PIN);
    SD.begin();
    OLED.begin();
    initialCO2 = getBaselineCO2(); // Capture baseline CO₂ level
}

void loop() {
    currentCO2 = CO2Sensor.readCO2();
    ratio = currentCO2 / initialCO2;

    // Display current CO₂ and ratio
    OLED.displayCO2(currentCO2);
    OLED.displayRatio(ratio);

    // Log data to SD card
    log data(currentCO2, ratio);

    // Alert if the ratio exceeds the threshold
    if (ratio >= RATIO_THRESHOLD) {
        OLED.displayAlert("Spoiling");
    }

    delay(600000); // Check every 10 minutes
}

float getBaselineCO2() {
    // Function to average CO₂ readings over a few minutes
```

```
    float sum = 0;
    for (int i = 0; i < BASELINE_INTERVAL; i++) {
        sum += CO2Sensor.readCO2();
        delay(60000); // 1-minute delay for each reading
    }
    return sum / BASELINE_INTERVAL;
}

void log data(float co2, float ratio) {
    // Function to log data to SD card
    SD.write("CO2: " + String(co2) + ", Ratio: " + String(ratio));
}
```

7. Analysis and Final Threshold Setting

- **Initial Tests:** After testing, analyze the recorded CO_2 ratios for each vegetable when spoilage becomes visually or olfactorily evident.
- **Set Thresholds:** Determine an average ratio threshold for spoilage based on your tests. This threshold will serve as the point at which the device triggers an alert, indicating likely spoilage.
- **Experiment with Thresholds:** CO_2 emission rates can vary, so be prepared to fine-tune your threshold ratio based on observed data.
- **Temperature and Humidity Impact:** These can affect CO_2 emissions. Test under conditions similar to a household fridge or room temperature if possible.
- **Reusable Design:** Since the sensor will monitor gases and not touch the food, this setup can be easily reused across different samples without contamination issues.

Code for Displaying Freshness Status on an OLED Display

This version displays either "Fresh" or "Spoiling" based on the CO_2 ratio.

```
#include <CO2SensorLibrary.h>   // Hypothetical library for your CO₂ sensor
#include <Adafruit_SSD1306.h>   // Library for OLED display (Adafruit SSD1306)

#define SCREEN_WIDTH 128        // OLED display width, in pixels
#define SCREEN_HEIGHT 32        // OLED display height, in pixels
#define OLED_RESET -1           // Reset pin (set to -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define CO2_SENSOR_PIN A0
#define RATIO_THRESHOLD 1.5     // Placeholder threshold; adjust after data collection

float initialCO2 = 0;
float currentCO2 = 0;
float ratio = 0;

void setup() {
```

```
    Serial.begin(9600);

    // Initialize CO₂ sensor
    CO2Sensor.begin(CO2_SENSOR_PIN);

    // Initialize OLED display
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Check OLED connection at I2C address
0x3C
        Serial.println(F("OLED display failed to initialize"));
        for (;;);  // Infinite loop if display fails
    }

    display.clearDisplay();    // Clear initial display
    display.setTextSize(1);    // Set text size to fit status messages
    display.setTextColor(SSD1306_WHITE);  // Set color of the text
    display.display();

    // Set initial baseline CO₂ level
    initialCO2 = getBaselineCO2();  // Function to capture baseline CO₂
}

void loop() {
    // Measure current CO₂ and calculate the ratio
    currentCO2 = CO2Sensor.readCO2();
    ratio = currentCO2 / initialCO2;

    // Display the current CO₂ and freshness status on OLED
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print("CO2 Level: ");
    display.println(currentCO2);
    display.print("Ratio: ");
    display.println(ratio);

    // Check if the food is spoiling
    if (ratio >= RATIO_THRESHOLD) {
        display.setCursor(0, 16);
        display.println("Status: Spoiling");
    } else {
        display.setCursor(0, 16);
        display.println("Status: Fresh");
    }

    // Update the display with the new information
    display.display();

    delay(600000);  // Wait 10 minutes before checking again
}

// Function to get initial baseline CO₂ level
float getBaselineCO2() {
    float sum = 0;
    int readings = 10;  // Number of readings to average for baseline
```

```
    for (int i = 0; i < readings; i++) {
        sum += CO2Sensor.readCO2();
        delay(60000);  // Wait 1 minute between readings
    }
    return sum / readings;  // Return average as baseline
}
```

Explanation of Key Sections

1. **Initialize and Setup the OLED Display**
    - This code uses the **Adafruit SSD1306** library, a common library for OLED displays.
    - We specify the display width and height and initialize it with the `display.begin()` function. If it fails, the code will print an error message to the serial monitor.
2. **Display Current CO_2 Levels and Freshness Status**
    - Every loop, the code reads the current CO_2 level, calculates the ratio, and updates the display.
    - The `display.clearDisplay()` function clears the previous display so you can print the updated information.
    - The code then shows the **CO_2 level**, **CO_2 ratio**, and a **status message** (either "Fresh" or "Spoiling") based on the threshold ratio.
3. **Threshold Check for Spoiling Status**
    - If `ratio >= RATIO_THRESHOLD`, the display shows "Status: Spoiling," indicating that the food may be going bad.
    - If the ratio is below the threshold, the display will show "Status: Fresh."

Add other display effects if you want to make it clearer, like flashing the "Spoiling" message when the threshold is crossed:

- **Flash the Message:** Alternate between displaying and clearing the message every second if the food is spoiling. You can add a loop to flash the "Spoiling" message for a few seconds.

LED Alert Light: Have the LED light up (or blink) when `ratio >= RATIO_THRESHOLD`. This would require just a couple of additional lines to turn on the LED:

```
#define ALERT_LED_PIN 13   // Connect LED to digital pin 13 (or another available pin)

void setup() {
    pinMode(ALERT_LED_PIN, OUTPUT);  // Set LED pin as output
    // Other setup code here...
}
```

```
void loop() {
    // Other loop code here...

    // Turn on LED if food is spoiling
    if (ratio >= RATIO_THRESHOLD) {
        digitalWrite(ALERT_LED_PIN, HIGH);   // Turn on LED
    } else {
        digitalWrite(ALERT_LED_PIN, LOW);    // Turn off LED
    }

    // Delay and loop...
}
```

Testing and Calibration

Before you finalize, be sure to:

- **Test the Threshold** by checking CO_2 levels and ratios for the initial vegetables.
- **Calibrate the Threshold Value** (`RATIO_THRESHOLD`) based on your tests.

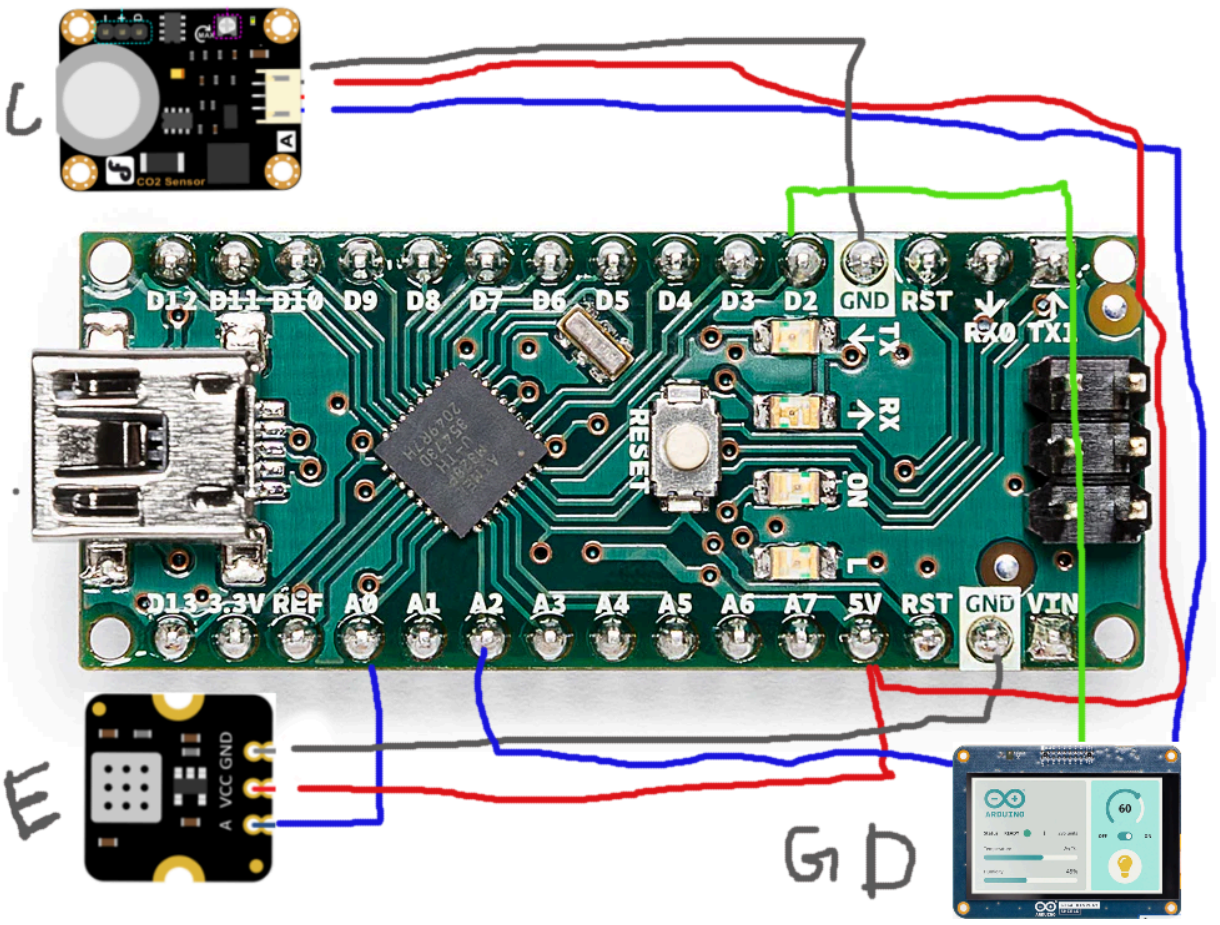
Nov 7

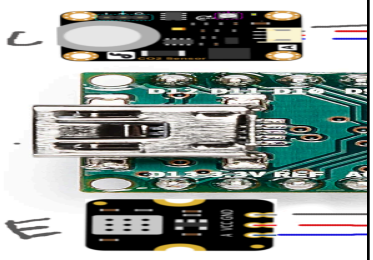
- New sensor (Arduino compatible)
    - https://store-usa.arduino.cc/products/gravity-analog-co2-gas-sensor-mg-811-sensor
    - Sensor wiki: https://wiki.dfrobot.com/CO2_Sensor_SKU_SEN0159
- Reference project (Sensor station)
    - https://projecthub.arduino.cc/cstram/sensor-station-aff0d6
- More Fruit Fly info: https://www.caltech.edu/about/news/how-fruit-flies-sniff-out-their-environments
- Updated Gantt Chart
    - Gantt chart
- GIGA DISPLAY
    - https://www.amazon.com/Arduino-GIGA-Display-Shield-ASX00039/dp/B0CKBZLTND?crid=1059IK5AHHAYD&dib=eyJ2IjoiMSJ9.htia3oB7nDohXjKabCq8Fl7ALX1GlKfaTh_0-DqNyYSE4Fbqi1JEa2GbKIYAS91kQxcOBcdHe7jp77EcoaMevHR_bG91PduhxiXtSshuK2_00JYF_96Gu0Bb-s_oY4KjCGuJX9O6K1loZr32O2QT5mDQb1kja_h2aC0DIpnfh37O5VuHycJo0-64Myr9rFQkKycQ0Tee8RcBL4pJ9cmzHqlvvRoJOvLmczPn_PF_BQY.ZKPAwJI8l33he0LsGr-_jJE-GXSx3mq1ANa9z8axl9M&dib_tag=se&keywords=OLED+Arduino+GIGA&qid=1731009952&sprefix=oled+arduino+giga%2Caps%2C78&sr=8-2
- Scale

    - https://projecthub.arduino.cc/mPelectronic/high-precision-scale-with-arduino-b9daee
        - https://www.amazon.com/DIYmall-Weighing-Conversion-Sensors-Microcontroller/dp/B010FG9RXO
    - https://forum.arduino.cc/t/how-do-i-use-hx711-and-a-load-cell/1251675
    - https://randomnerdtutorials.com/arduino-load-cell-hx711/
        - Load cell
            - https://www.sparkfun.com/products/14727

Nov 14

- Configured the arduino and the sensors connectivity. We decided to use a giga-display to reflect the sensor readings. As of now we are in between making it a color display, as in green for fresh and red for rotting, and, text. We want it to be as simple as possible so we won't display the actual reading but the implications instead.
    - Schematic:



	CO2 within healthy range	CO2 exceeds critical value
Ethanol within healthy range	Return  Fresh	Return Eat Soon
Ethanol exceeds critical value	Return Rotten	Return Rotten

Nov 25 (Final Report Breakdown)

- Worked on Prototype Breakdown Plan
    - Prototype Breakdown Plan
- Split up roles for final report
    - Intro and problem description (Zehra, Zeshui, Christopher)
        - Broadly describe the goal of your project
        - Include a quantitative description of the problem (include your sources!)
        - Who are the stakeholders? What do they want out of the solution?
        - Include relevant prior art in both engineering and biology spaces. This should be a summary of the background research report you submitted (with any relevant updates).
        - Include your problem statement. It should be concise (2-3 sentences) and follow the guidelines we discussed
        - Summarize your team's approach to the project.
        - Add a transition at the end of this section that guides the reader through what will be discussed in the rest of the report and makes them want to read it.
    - Design and build (everyone)
        - What is your approach to the problem? How did you bring the biology into the engineering?
        - What are your success metrics? They should be quantitative and justified! Why are these your metrics?
        - What principles guided your prototypes? How did you choose your design(s)?
        - What decisions/assumptions/simplifications did you make to your design?
        - What prototypes did you make? This includes mathematical models, computer models, and physical prototypes.
        - How did you create your prototypes? Why did you choose these methods?
    - Testing and results (Dorming: Allen, Ahikara, Somin)
        - How did you test your prototypes? Include the relevant details of your testing procedure
            - Ask ~20 rooms in the dorms, "Do you have any food in your fridge that you think is going bad/bad?" and use our prototype on the food. If the food is not going bad, tell them (we've prevented food waste!). Record the number/proportion of rooms that we've prevented food waste for & didn't - then we have an approximate statistic of how much our prototype helped our consumers.

        - What results did you get?
        - What do those results mean? How do they compare to your success metrics?
        - How did you adjust your design in response to your results?
    - Conclusion (Dorming: Allen, Ahikara, Somin)
        - Summarize the findings in your report. Did you hit your success metrics? Why or why not?
        - What is the bigger picture of your work? What do your results mean for your problem?
        - Include future work. What are the following directions the research should take? What would
        - you do if you had more time?
        - What ethical dimensions do you consider or need to consider moving forward?
        - End by motivating your project – make the reader want to read your following report!
    - Process reflection (Will be done collectively at the end) (Dorming: Allen, Ahikara, Somin)
        - What was this project like for you and your team?
        - What did you learn?
        - What did you do well? What would you do differently?
        - What skills will you take with you for the rest of your engineering career?
- Practice final showcase (**Assume no prior knowledge from your audience.**)
    - Canva link: https://www.canva.com/design/DAGXhOOKCvY/M1WF1583cLFv0MpcnFBN7w/edit?utm_content=DAGXhOOKCvY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
    - Very brief introduction and motivation.
    - Include the relevant details of guiding design principles
    - Design choices and decisions, as well as your results and analysis.
    - Hone a 2-sentence elevator pitch to catch the viewers' attention immediately.

Nov 26

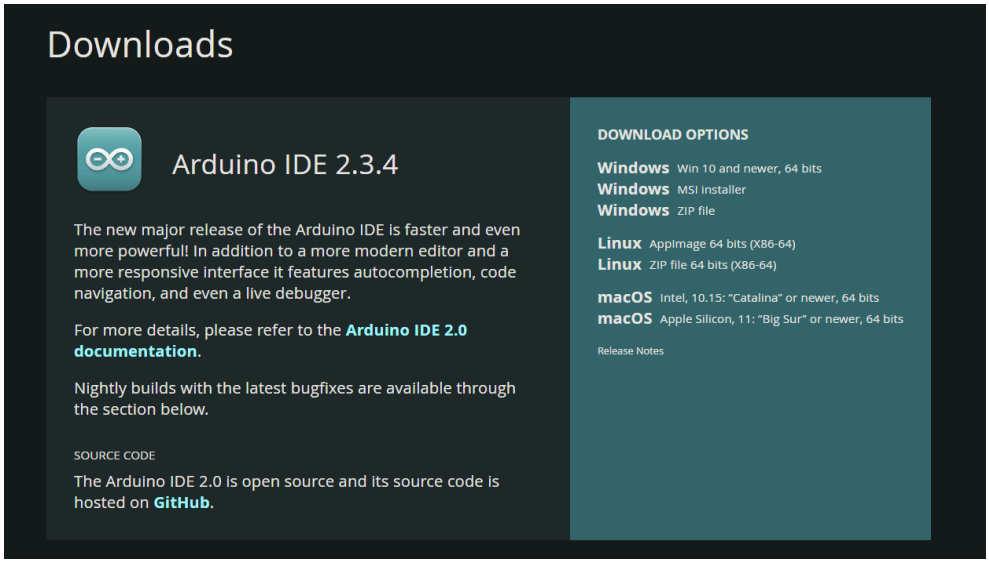
- Ethanol sensor
  https://projecthub.arduino.cc/sheekar/mq3-alcohol-sensor-with-arduino-sheekar-banerjee-2d79b8
- CO2 Sensor
  https://wiki.dfrobot.com/CO2_Sensor_SKU_SEN0159
  CAD model: https://grabcad.com/library/sen0159-1

https://media.digikey.com/pdf/data%20sheets/dfrobot%20pdfs/sen0159_web.pdf

```
const int AOUTpin=0;//the AOUT pin of the alcohol sensor goes into analog  pin A0 of the Arduino
const int DOUTpin=8;//the DOUT pin of the alcohol sensor  goes into digital pin D8 of the Arduino

int limit;
int value;

void  setup() {
Serial.begin(115200);//sets the baud rate
pinMode(DOUTpin, INPUT);//sets  the pin as an input to the Arduino
pinMode(ledPin, OUTPUT);//sets the pin as  an output of the Arduino
}

void loop()
{
value= analogRead(AOUTpin);//reads  the analog value from the alcohol sensor's AOUT pin
limit= digitalRead(DOUTpin);//reads  the digital value from the alcohol sensor's DOUT pin
Serial.print("Alcohol value:  ");
Serial.println(value);//prints the alcohol value
Serial.print("Limit:  ");
Serial.print(limit);//prints the limit reached as either LOW or HIGH (above  or underneath)
delay(500);
}
```

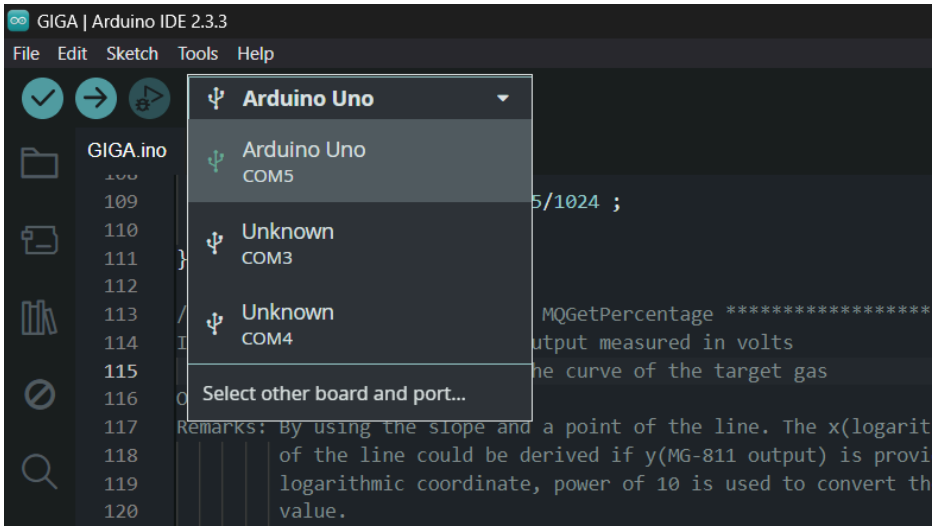
Dec 5th (INSTRUCTIONS ON HOW TO GIGASNIFF)

- Download Arduino IDE
    - https://www.arduino.cc/en/software

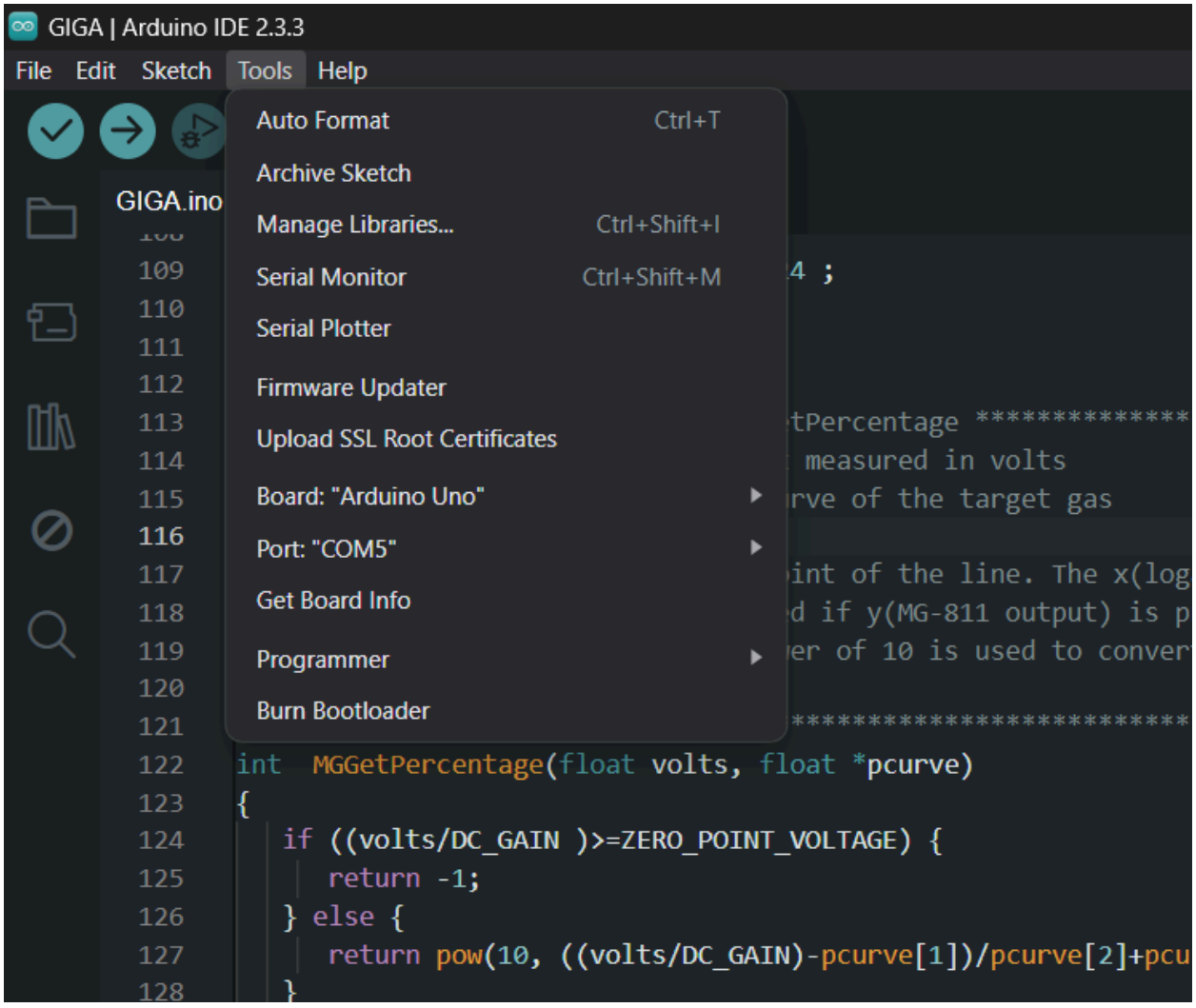


- Download this file from the shared team folder.

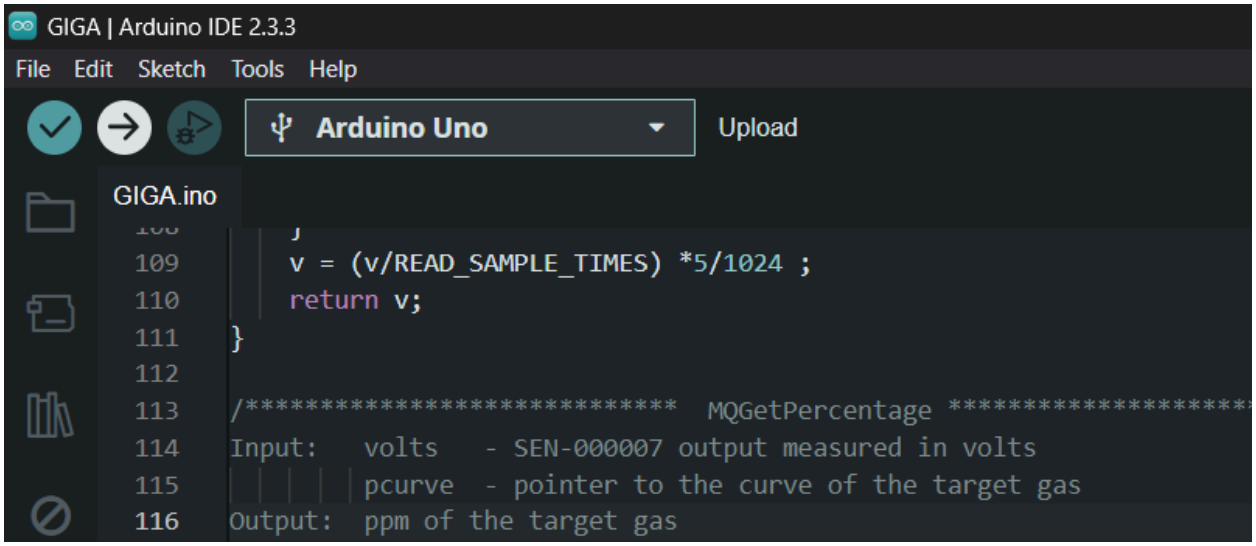
- Open file in IDE
- After connecting Arduino via USB, pick the one that says Arduino Uno.



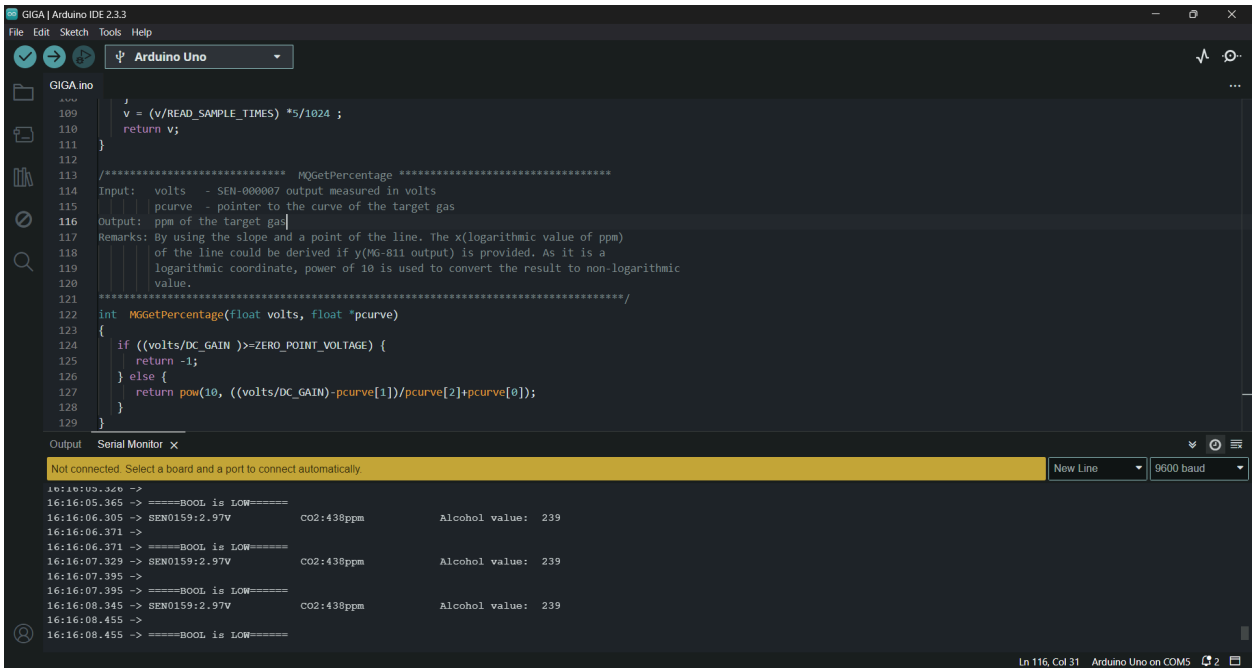
- Under tools, start the serial monitor.



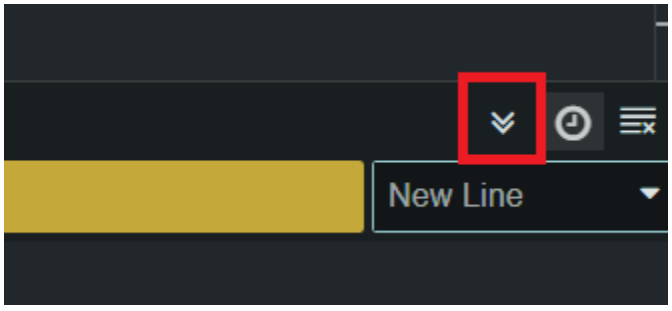
- Click the arrow to upload the code to Arduino.



- You should see the output from the sensor in the serial monitor.



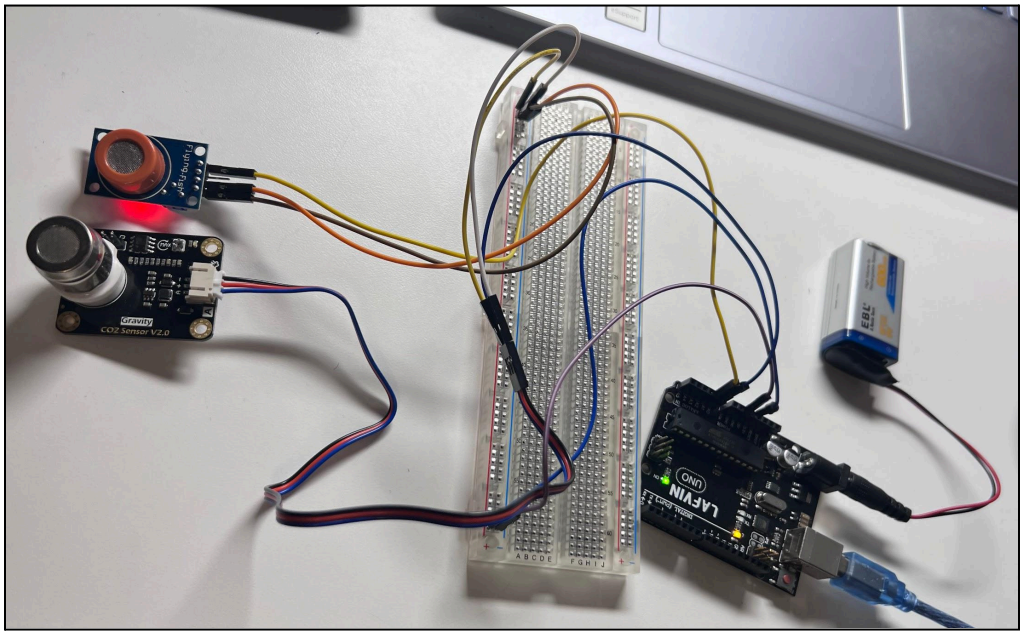
- The newest code will be at the bottom, so you will NOT need to scroll down
- ENABLE AUTO SCROLL



- NORMAL CO2 LEVELS: 400-460
- NORMAL ETHANOL LEVELS: 200-300

- For data collection, pick the highest value you see while GIGASNIFFING

Dec 6th - day 1 of testing (After 1 day of rot)



Setup of the wiring
Orange sensor - ethanol sensor
Black/white sensor - CO2 sensor
Connected to battery
Connected to Allen's laptop (Arduino IDE)
During Calibration:

```
=====BOOL is HIGH======
SEN0159:2.47V           CO2:469ppm           Alcohol value:  103
```

Standard:
CO_2:466ppm
Ethanol: 124.5 (ranged from 109 -119/136)

Raspberry - Co2: 471-474 (472.5) Ethanol: 119-132 (127)	Banana - Co2: 474 Ethanol: 186-195 (190.5)	Pear - Co2: 471-474 (472.5) Ethanol: 121-163, (147)

Dec 7th:

Standard values:

CO_2: 453 - 460 ppm

Final: 456.5 ppm

Ethanol: 94 - 113 ppm

Final: 103.5 ppm

Raspberries: Co_2: 466/467 - 473 (470) Ethanol: 120	Banana: CO_2: 472/473 (472) Ethanol: 190.5	Pear: CO_2: 471-473 (471) Ethanol: 253-260 (256)
		

Dec 8th:

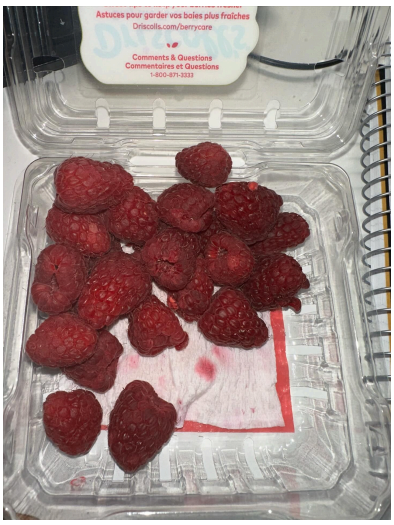
Standard:
CO_2: 468/469 ppm
Final: 468.5 ppm
Ethanol: 105 ppm

Raspberries: Co2: 478 Ethanol: 184/185 (184.5)	Banana: CO_2: 469-471 (469) Ethanol: 340	Pear: Co2: 480-484 (483) Ethanol: 280-287 (282)
		

Dec 9th:

Standard:
CO_2: 479
Alcohol: 108

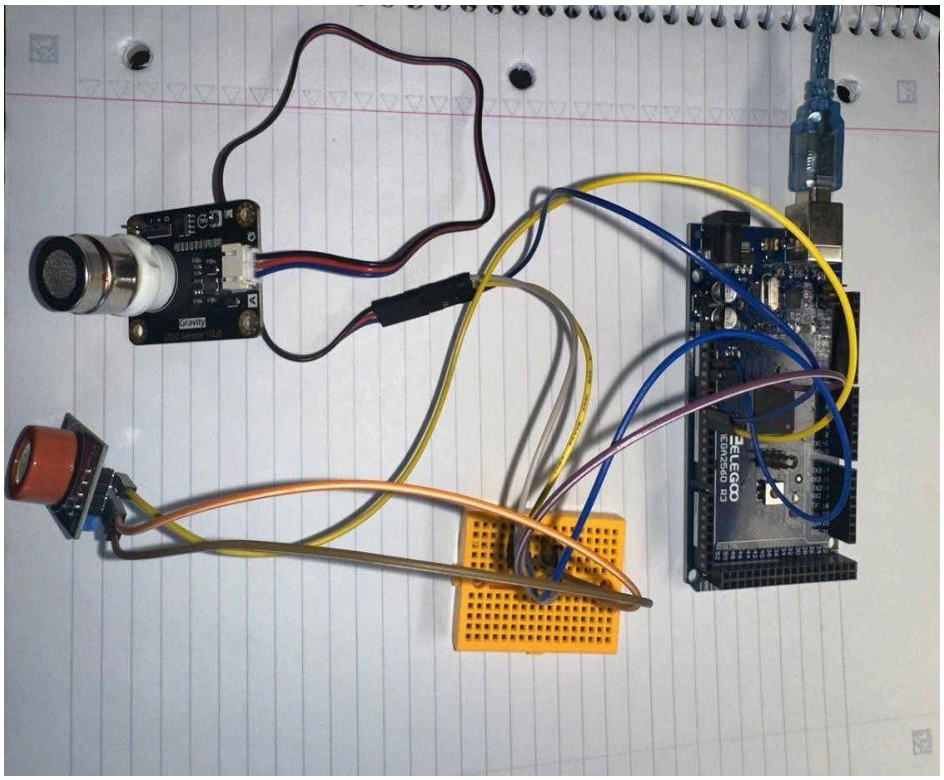
Raspberries: Co2: 511-513 (512) Ethanol: 280-283 (282)	Banana: CO_2: 510-512 (512) Ethanol: 398-401 (401)	Pear: Co2: 506 Ethanol: 294-295 (294)

Dec 10th:

Standard:
CO_2: 480
Alcohol: 168

Raspberries: Co2: 504-506 Final: 505 Ethanol: 340 Final: 340	Banana: CO_2: 502/503 Final: 503 Ethanol: 480-483 Final: 480	Pear (Very Mushy): Co2: 496-497 Final: 496 Ethanol: 224-232 Final: 229
		



- We then upgraded from the Arduino Uno to the Mega due to insufficient on-board memory, which was causing our code to crash. The second iteration of the code calculates an average of the CO_2 and ethanol readings and compares the current levels to this average. If the current levels exceed a defined threshold, the system will indicate high levels of CO_2 and ethanol, respectively.

Dec 11th:

Standard:
CO_2: 492
Alcohol: 180-181
Final: 180.5

Raspberries: Co2: 530-539 Final: 525 Ethanol: 327 -389 Final: 375	Banana: Final: 527 Ethanol: 215 - 263 Final: 236	Pear (Very Mushy): Co2: 510 Final: 495 Ethanol: 258 Final: 275

Dec 12th:

Standard:
CO_2: 498.5
Alcohol: 118

Raspberries: Co2: 513-527 Final: 523 Ethanol:307-395 Final: 336	Banana: Co2: 504-523 Final: 511 Ethanol: 249-295 Final: 285	Pear (Last Day): Co2: 519-525 Final: 521 Ethanol: 195-244 Final: 233
		

Dec 13th:

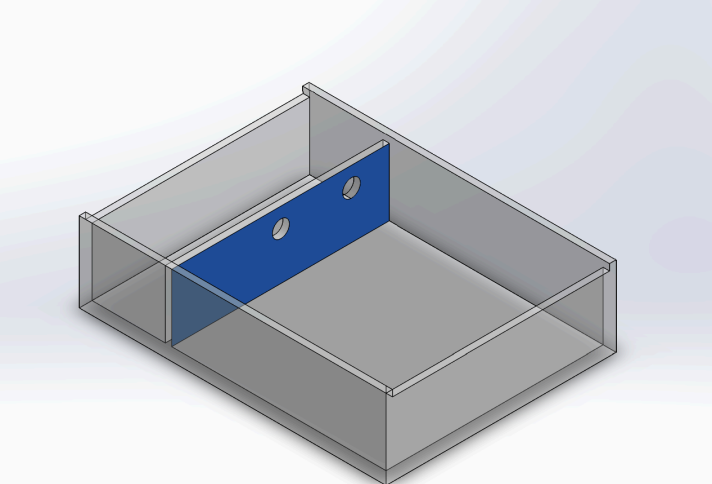
Standard:
CO_2: 500
Alcohol: 146-199
Final: 172.5

Raspberries (Last day): Co2: 545 Ethanol: 312-367 Final: 356	Banana: Co2: 525-541 Final: 527 Ethanol: 281-327 Final: 315	Pear: Visibly rotten, no longer tested
		DISPOSED

- Our threshold values were determined using the data collected above.
  - There were limited online sources offering concrete values for carbon dioxide and ethanol emitted by the fruits we were testing, so we utilized the trends in our data.
  - THRESHOLD VALUE FOR CO_2 : 25 ppm
  - THRESHOLD VALUE FOR ETHANOL: 100 ppm

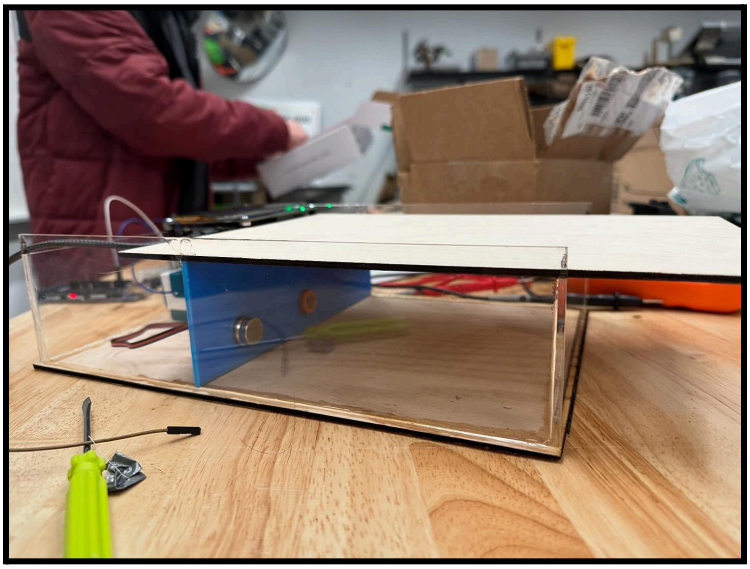
Dec 14th:

- At this point, we have started building our box to hold the device:



- The box is meant to be entirely acrylic, but due to the limited material in the maker space, some components had to be cut from wood.
- The smaller compartment is intended to hold the giga display and electronics, while the divider with the holes is intended to allow the sensors to poke through.
- The larger component holds the item tested for freshness, and the box's lid is designed to slide on to ensure an airtight seal.
- The device was initially intended to be a wand. However, upon testing, we found that the carbon dioxide and/or ethanol emitted from the produce mixed into the air too quickly to make a measurable difference.
- This box design allows for more accurate readings and remains an appropriately sized kitchen appliance.

FINAL PRODUCT:



- As the right image shows, the sensors poke through the blue divider, and the electronics are in the smaller compartment.

Dec 17th:

- Arduino GIGA (WIFI), compatible with our giga display purchased earlier in the semester, arrived.
- Below is the code that we wrote to configure the giga display:

```
for (int i = 0; i < BUFFER_SIZE; i++) {
        display.setTextColor(WHITE);
        display.print(" "); // Adjust spacing for alignment
        display.print(i);
        display.print("      "); // Adjust spacing for alignment
        display.print(co2Readings[i]); // Print CO2 reading
        display.print("      "); // Adjust spacing for alignment
        display.println(alcoholReadings[i]); // Print Alcohol reading

        int currentX = display.getCursorX();
        int currentY = display.getCursorY();
        display.setCursor(currentX, currentY + lineHeight);
    }

    display.setCursor(350, 0);
    display.setTextColor(WHITE);
    display.print("Buffer Index: ");
    display.print(bufferIndex); // Print Buffer Index
    display.print("   A: ");     // Label for A
    display.println(A);          // Print A value

    display.setCursor(350, 50);
    display.print("SEN0159: ");
    display.print(volts, 2); // 2 decimal places for voltage
    display.print("V");
    display.setCursor(350, 100);
    display.print("CO2: ");
    display.print(co2Readings[lastIndex]);
    display.print(" | Alcohol: ");
    display.println(alcoholReadings[lastIndex]);

    // Print averages
    display.setCursor(350, 150);
    display.print("Avg CO2: ");
    display.print(co2Average);
    display.print(" | Avg Alcohol: ");
    display.println(alcoholAverage);

    // Print alerts if deviations exceed thresholds
    if (co2Difference > CO2_DIFFERENCE_THRESHOLD) {
```

```
    display.setCursor(350, 200);
    display.setTextColor(RED);
    display.print("Hi CO2: ");
    display.print(co2Readings[lastIndex]);
    display.print(" ppm");
  }
  if (alcoholDifference > ALCOHOL_DIFFERENCE_THRESHOLD) {
    display.setCursor(350, 250);
    display.setTextColor(RED);
    display.print("Hi Alco: ");
    display.print(alcoholReadings[lastIndex]);
    display.print(" ");
  }
  // Control the LED based on the state
  if (rot) {
    display.setTextSize(4);
    display.setCursor(10, 350);
    rgb.on(255, 0, 0); // Red for rotten
    display.setTextColor(RED);
    Serial.println("Rotten: Red LED On");
    display.print("Rotten: Red LED On");
  }
  else if (soon) {
    display.setTextSize(4);
    display.setCursor(10, 300);
    rgb.on(255, 255, 0);
    display.setTextColor(YELLOW);
    Serial.println("Soon: Yellow LED On");
    display.print("Soon: Yellow LED On");
}
  else if (fresh) {
    display.setTextSize(4);
    display.setCursor(10, 300);
    rgb.on(0, 255, 0); // Green for fresh
    display.setTextColor(GREEN);
    Serial.println("Fresh: Green LED On");
    display.print("Fresh: Green LED On");
  }
  else {
    rgb.off();
  }
  Serial.println("-------------------------------------------");
  delay(500); // Short delay for readability in serial output
```

- This code follows the scheme we settled on November 14:
    - IF ETHANOL EXCEEDS THE THRESHOLD: RED

- IF CO_2 EXCEEDS THE THRESHOLD & ETHANOL DOESN'T: YELLOW
- IF BOTH ETHANOL AND CO_2 ARE BELOW THRESHOLD: GREEN

Dec 20th (New Beginnings)

- We finalized our final documentation and submitted it :)
- We finished our final report!
  - EID101B_Team 4_FinalReport
- Celebrated with KBBQ and said our goodbyes to Chris