Bus Vibrations

Presented by Liz Zhang and Max Winga
Department of Physics, University of Illinois at Urbana-Champaign
PHYS 398 DLP, April 29, 2022

Credit to Autumn Soliman, Daily Illini for the image

Project Prompt

“Characterization of surprisingly high frequency, large amplitude vibrations experienced by public transportation passengers. We saw signs of 0.5g, 120 Hz vibrations in the MTD bus seats. Yikes! Was that really happening?”

Methods

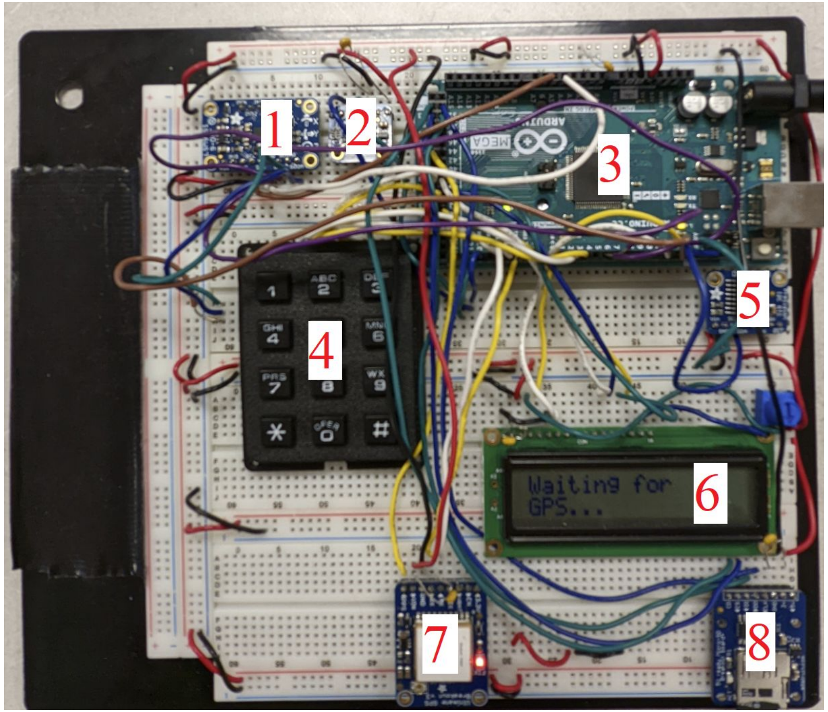
Arduino Device Goals

- Record acceleration in x, y, z
    - Read fast enough to analyze 100 Hz frequencies
- Record location, temperature, and speed
- Have necessary UI for control in the field
- Seperate devices need to synchronize data collected

Why use two devices?

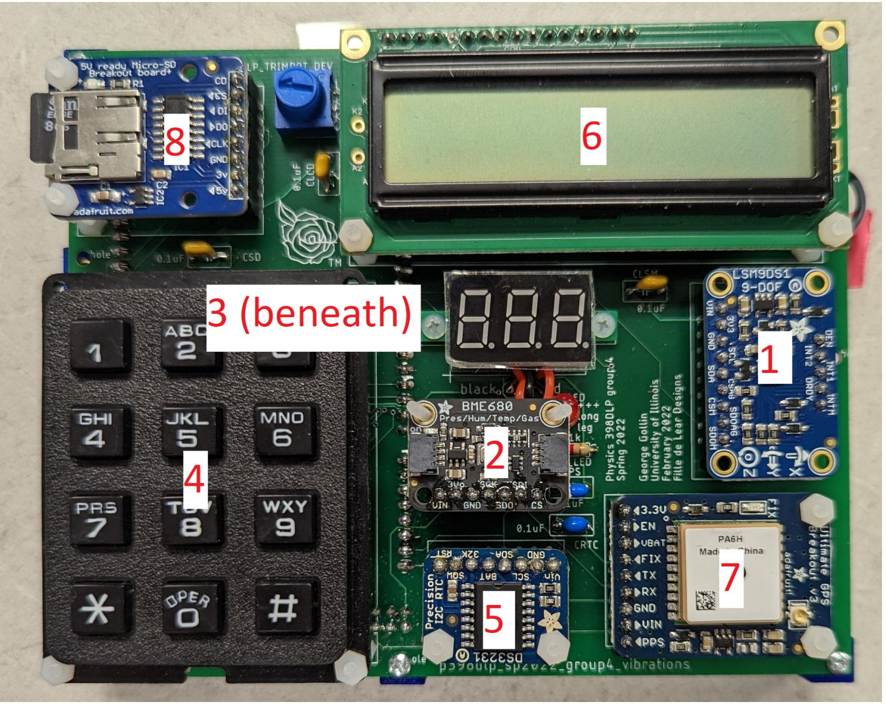
- Decision made early on
- Offload all non-accelerometer recording to separate board
- Increased the amount of time recording
- Eventually wasn’t necessary

Breadboard Layout



#	Component Name	Purpose of component
1	Adafruit LSM9DS1 Accelerometer	For vibration reading
2	Adafruit BME680	For temperature data
3	Adafruit MEGA 2560	Processor
4	Keypad	For function selection
5	Adafruit DS3231 Real Time Clock	For precise time tracking
6	LCD Display	For visual feedback
7	Adafruit Ultimate GPS Breakout v3	For GPS and time sync
8	Adafruit MicroSD Breakout	For data output

PCB Version



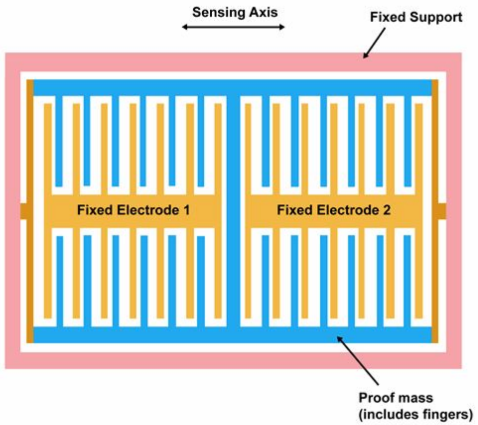
#	Component Name	Purpose of component
1	Adafruit LSM9DS1 Accelerometer	For vibration reading
2	Adafruit BME680	For temperature data
3	Adafruit MEGA 2560	Processor
4	Keypad	For function selection
5	Adafruit DS3231 Real Time Clock	For precise time tracking
6	LCD Display	For visual feedback
7	Adafruit Ultimate GPS Breakout v3	For GPS and time sync
8	Adafruit MicroSD Breakout	For data output

Note orientation of LSM9DS1

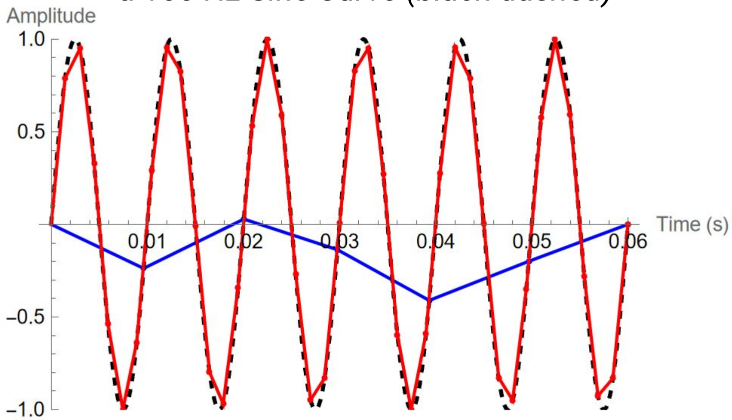
Adafruit LSM9DS1 Accelerometer

- MEMS accelerometer used for recording vibrations
- Disabled the gyroscope and magnetometer
- Used SPI vs. I2C
- Increased read speed from 119 Hz to 698 Hz

Diagram of MEMS Sensor



698 Hz (red) vs. 119 Hz (blue) Reading of a 100 Hz Sine Curve (black-dashed)



DAQ Code Overview

Unit A Goals:

- Record time stamped accelerometer data
- Be able to operate fully without computer interface
- Record extra necessary information for data analysis inside the CSV file

Unit B Goals:

- Record time stamped location data
- Record time stamped speed data
- Record time stamped temperature data
- Be able to operate fully without computer interface

Key Words

- **Datapoint**: Single reading of accelerometer
- **Cycle**: Collection of datapoints collected consecutively
- **Run**: Collection of cycles collected consecutively
- **Ride/day**: Collection of runs

Setup Functions

- Run setup functions for breakout boards
- Calibrate the RTC with the GPS
- Everything else done in loop

Synchronization

```
myFile.print("time, latitude, longitude, speed (mph), temp,");
while (digitalRead(GPS_PPS_pin) != 1) {}

currentTime = rtc.now();
if (currentTime.second() == 59) {
  if (currentTime.minute() == 59) {
    if (currentTime.hour() == 24) {
      futureHour = 1; futureMinute = 0; futureSecond = 0;
    } else {
      futureHour = currentTime.hour() + 1; futureMinute = 0; futureSecond = 0;
    }
  } else {
    futureHour = currentTime.hour(); futureMinute = currentTime.minute() + 1; futureSecond = 0;
  }
} else {
  futureHour = currentTime.hour(); futureMinute = currentTime.minute(); futureSecond = currentTime.second() + 1;
}
rtc_data = String(futureHour) + ":" + String(futureMinute) + ":" + String(futureSecond);
myFile.print(rtc_data + ",");

while (digitalRead(GPS_PPS_pin) != 1) {}
long millisStart = millis();
myFile.println(millisStart);
```

Unit A Loop

5 Functions: (selected via the keypad)

1. recordData()
2. calibrateAccelerometer()
3. displayInfo()
4. setDatapoints()
5. setCycles()

Keypad

Data Recording Loop

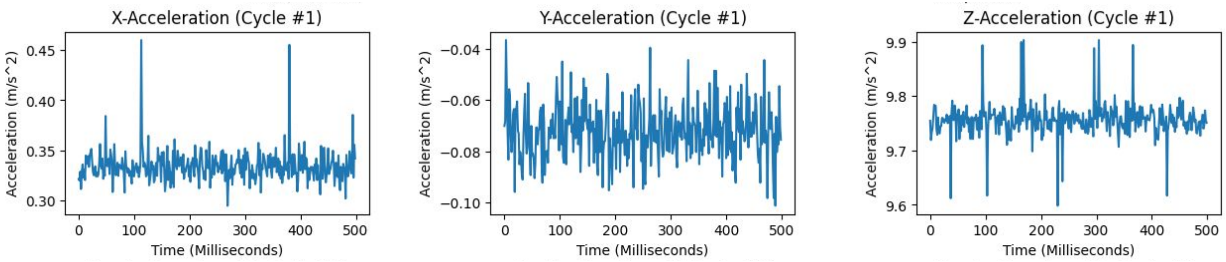
```
for (int c = 0; c < cycles; c++) {                  // Loop for set number of cycles
  millis1 = millis();                                // Save time right before reading begins
  for (int i = 0; i < datapoints; i++) {            // Loop for set number of datapoints
    lsm.read();                                      // Read the accelerometer
    lsm.getEvent(&a, &m, &g, &temp);                 // Get the data from the reading
    accel_data[i][0] = a.acceleration.x;             // Save the x value into the array
    accel_data[i][1] = a.acceleration.y;             // ..y
    accel_data[i][2] = a.acceleration.z;             // ..z
  }                                                  //
  millis2 = millis();                                // Save time right after reading ends
  //////////////////print to microSD//////////////////
  for (int j = 0; j < datapoints; j++) {
    x = accel_data[j][0] - x_off;                    // Adjust for linear offset
    y = accel_data[j][1] - y_off;                    // The offset is prerecorded and
    z = accel_data[j][2] - z_off;                    // saved in the x,y,z_off variables
    myFile.print(millis1); myFile.print(",");        // Print data into the CSV
    myFile.print(millis2); myFile.print(",");        // ..
    myFile.print(x, 6); myFile.print(",");           // ..
    myFile.print(y, 6); myFile.print(",");           // ..
    myFile.print(z, 6); myFile.println();            // ..
  }
```

Calibration

```
x = accel_data[j][0] - x_off;
y = accel_data[j][1] - y_off;
z = accel_data[j][2] - z_off;
```

- Offset from zero is linear
- Calibrate using gravity
- Store offset values in variables

Acceleration Data from Stationary LSM9DS1



Unit A CSV File

	A	B	C	D	E	F	G	H
1	millis1	millis2	x_accel	y_accel	z_accel	30	300	18:18:12
2	3527540	3527970	0.240568	-1.116272	9.381929			
3	3527540	3527970	0.184337	-1.048076	9.320912			
4	3527540	3527970	0.089821	-1.147378	9.33168			
5	3527540	3527970	0.193908	-1.098325	9.375946			
6	3527540	3527970	0.154427	-1.154557	9.402268			
7	3527540	3527970	0.092213	-1.207199	9.879636			
8	3527540	3527970	0.205872	-1.256252	9.889208			
9	3527540	3527970	0.238176	-1.226341	9.884422			
10	3527540	3527970	0.269282	-1.194038	9.8916			
11	3527540	3527970	0.350638	-1.354357	9.885618			
12	3527540	3527970	0.384138	-1.318465	9.925099			
...								
300	3527540	3527970	0.247747	-1.615175	10.045937			
301	3527540	3527970	0.105374	-1.506302	10.11174			
302	3530185	3530615	0.703579	-1.441696	9.952617			
303	3530185	3530615	0.616241	-1.50032	9.879636			

Unit B Loop

- Only one function
- Initializes some things and waits for GPS data
- Upon data received writes time, latitude, longitude, speed, and temperature
- Stops by holding the ‘#’ key
- We have had a multitude of issues with the GPS data

Unit B CSV File

	A	B	C	D	E
1	time	latitude	longitude	speed (mph)	temp
2	18x39x08	4006.2519	-8813.6611	7.24	20.1
3	18x39x09	4006.2524	-8813.6533	22.94	20.11
4	18x39x10	4006.2531	-8813.6464	22.94	20.15
5	18x39x11	4006.2539	-8813.6396	21.61	20.19
6	18x39x12	4006.2543	-8813.6328	21.61	20.24
7	18x39x13	4006.2546	-8813.6259	21.62	20.27
8	18x39x14	4006.2548	-8813.6191	21.62	20.3
9	18x39x15	4006.2553	-8813.6123	21.11	20.33
10	18x39x16	4006.2556	-8813.6054	21.11	20.37
11	18x39x17	4006.2556	-8813.5986	20.81	20.4
12	18x39x18	4006.2556	-8813.5927	20.81	20.43
13	18x39x19	4006.2556	-8813.5859	20.31	20.45

Other Variables Noted

- Bus Type
- Weather
- Device Location
- Notes on events

```
Date: 3/28/22
Start Time: 12:20pm
Runs: 15
Bus Number: 1169
Bus Type: Regular length hybrid
Weather: Clear, Dry, Cold
Location: Back floor middle, Front left middle
Run Notes: R1-R2 normal motion (30 cycles),
R3 increase to 70 cycles, R4 normal motion,
R5 something buzzy turned on at the front of the bus just for the beginning,
R6 buzzy in the beginning as well, R7 normal motion,
R8 passed a donut shop, R9-R10 normal motion,
R11 moved to front left middle increased cycles to 200 (gps data in two files
R12 another long boi nothing to note, R13 changed down to 100 cycles (stoppec
R14-15 cycles down to 30
```

MTD Collaboration

- Initial interview
  - Bus engine/suspension
  - Possible causes of vibrations
- Data collection
  - Diesel vs. Hydrogen comparison
  - Speed testing with Hybrid

Results

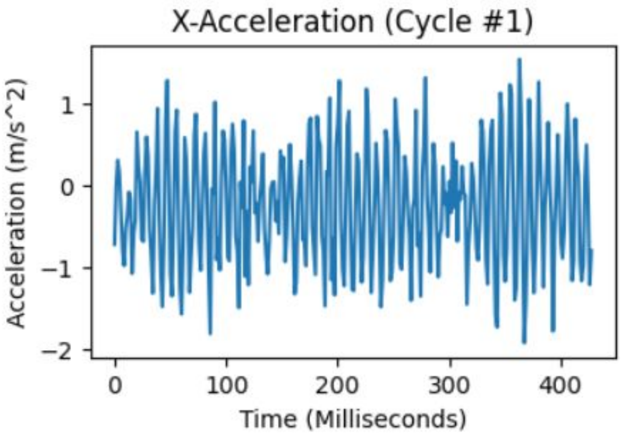
How We Interpret Our Data

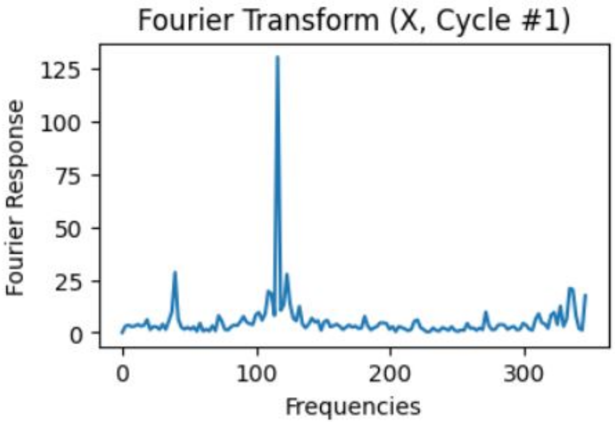
- Fourier transforms
- Root-mean-square deviation (RMSD)
- Comparing versus different variables
    - Bus type
    - Speed
    - Location
    - Temperature

Fourier Transforms

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \omega} dt$$

- Converts real signals to frequency space
- Very useful when investigating oscillations





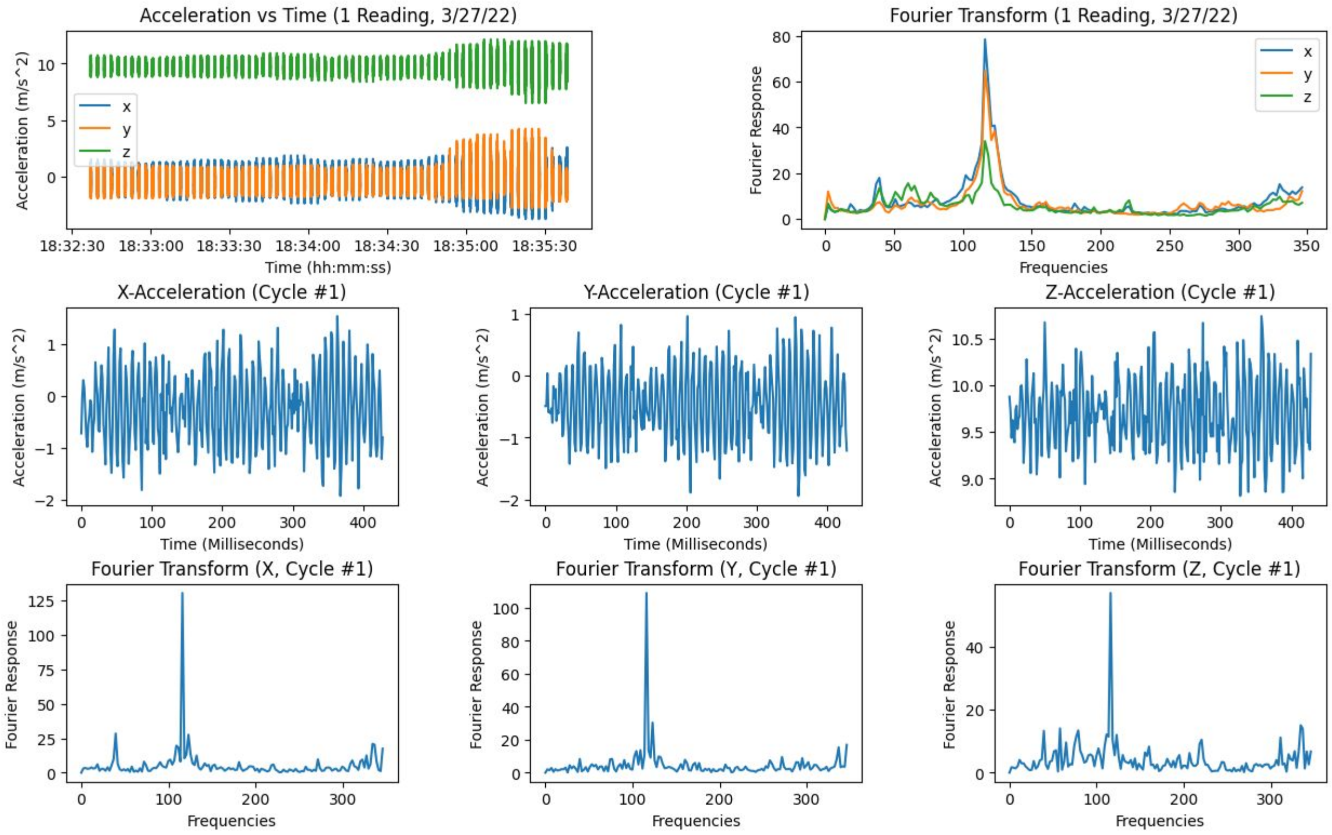
Root-Mean-Square Deviation (RMSD)

- How much the sample deviates from the mean
- Useful for comparing the amplitude of oscillations
- Same as standard deviation

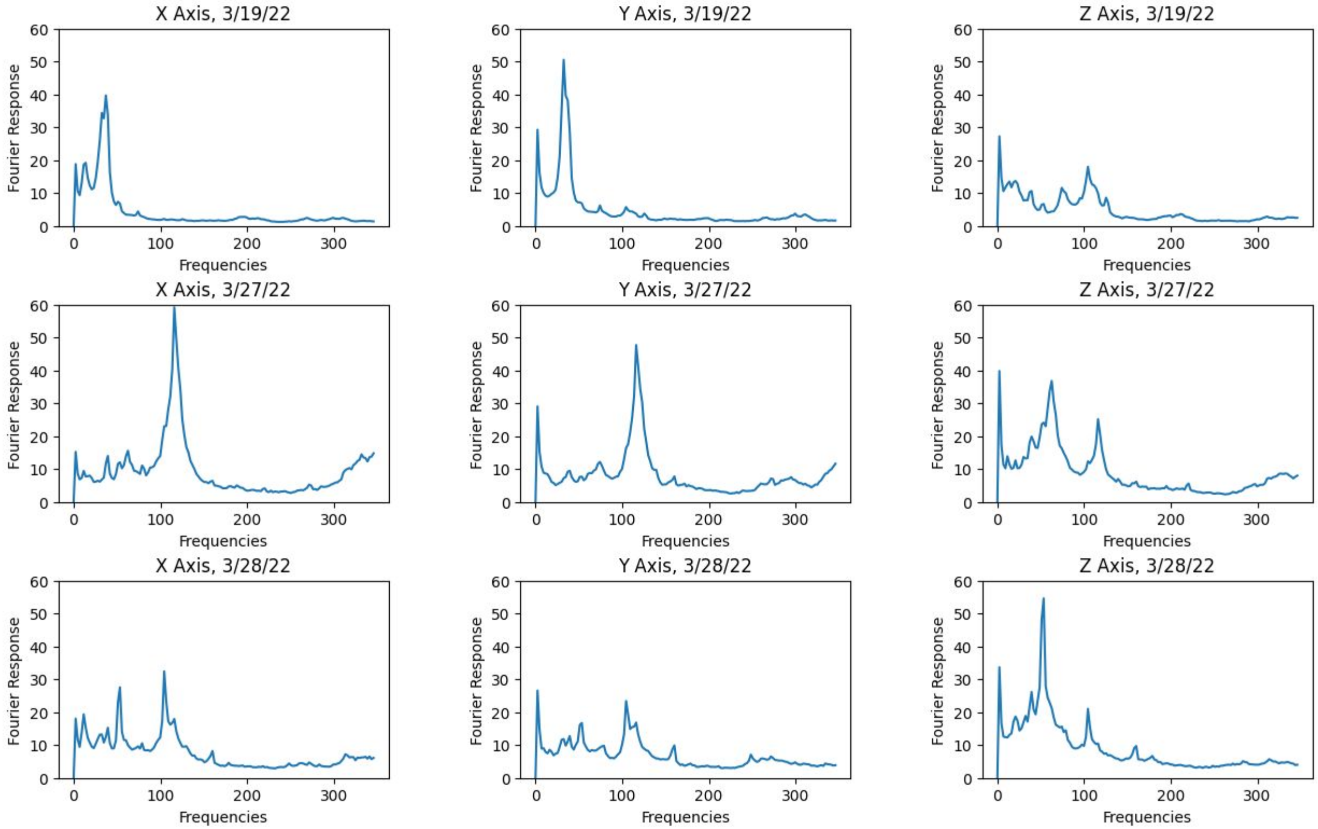
$$RMSD_{cycle} = \sqrt{E[a^2] - E[a]^2}$$

$$Where\ E[a] = \frac{1}{n}\sum_{n} a_n\ ,\ \ E[a^2] = \frac{1}{n}\sum_{n} a_n^2\ ,\ \ 1 \leq n \leq datapoints\ in\ cycle$$

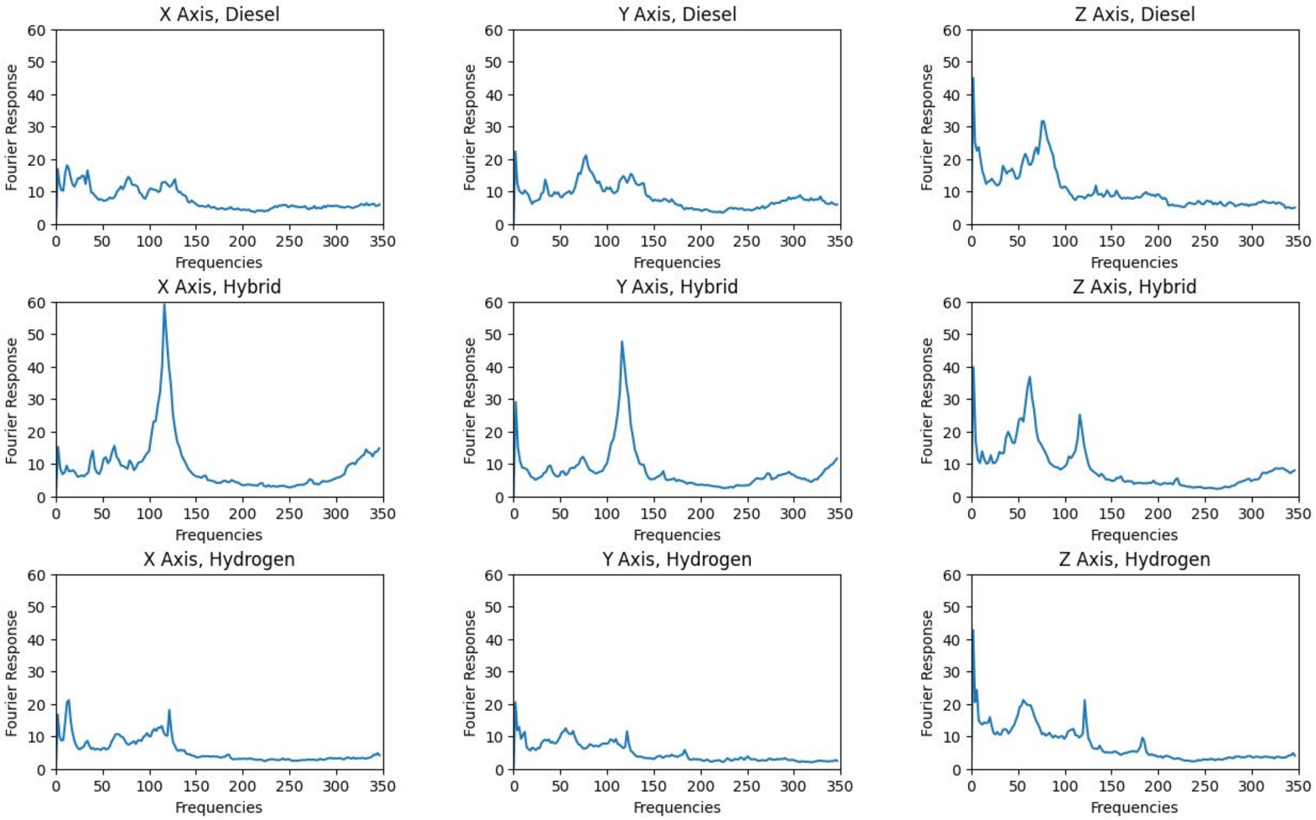
Sample Acceleration Data and Fourier Response, 3/27/22



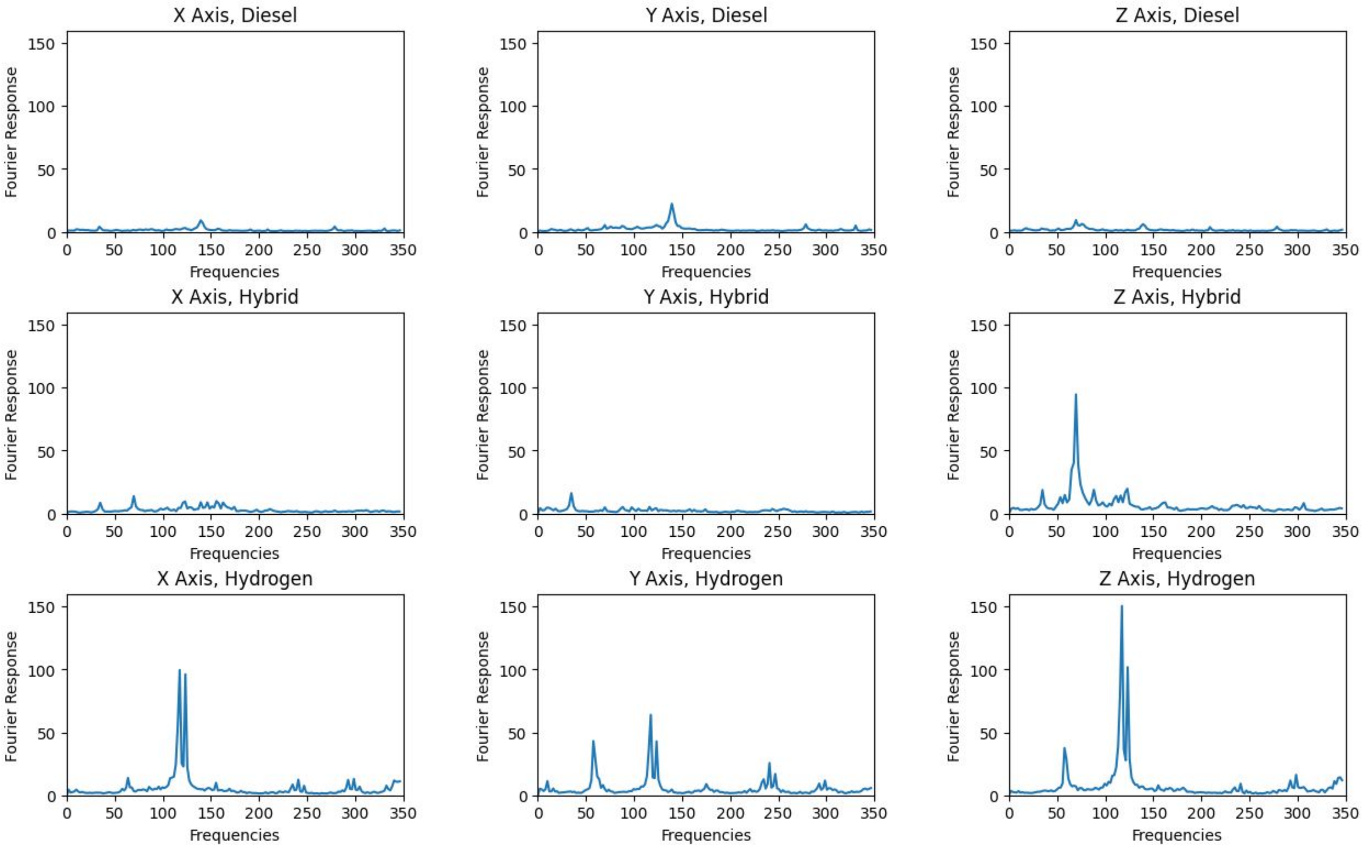
Over Fourier Transform Comparison - Across Days



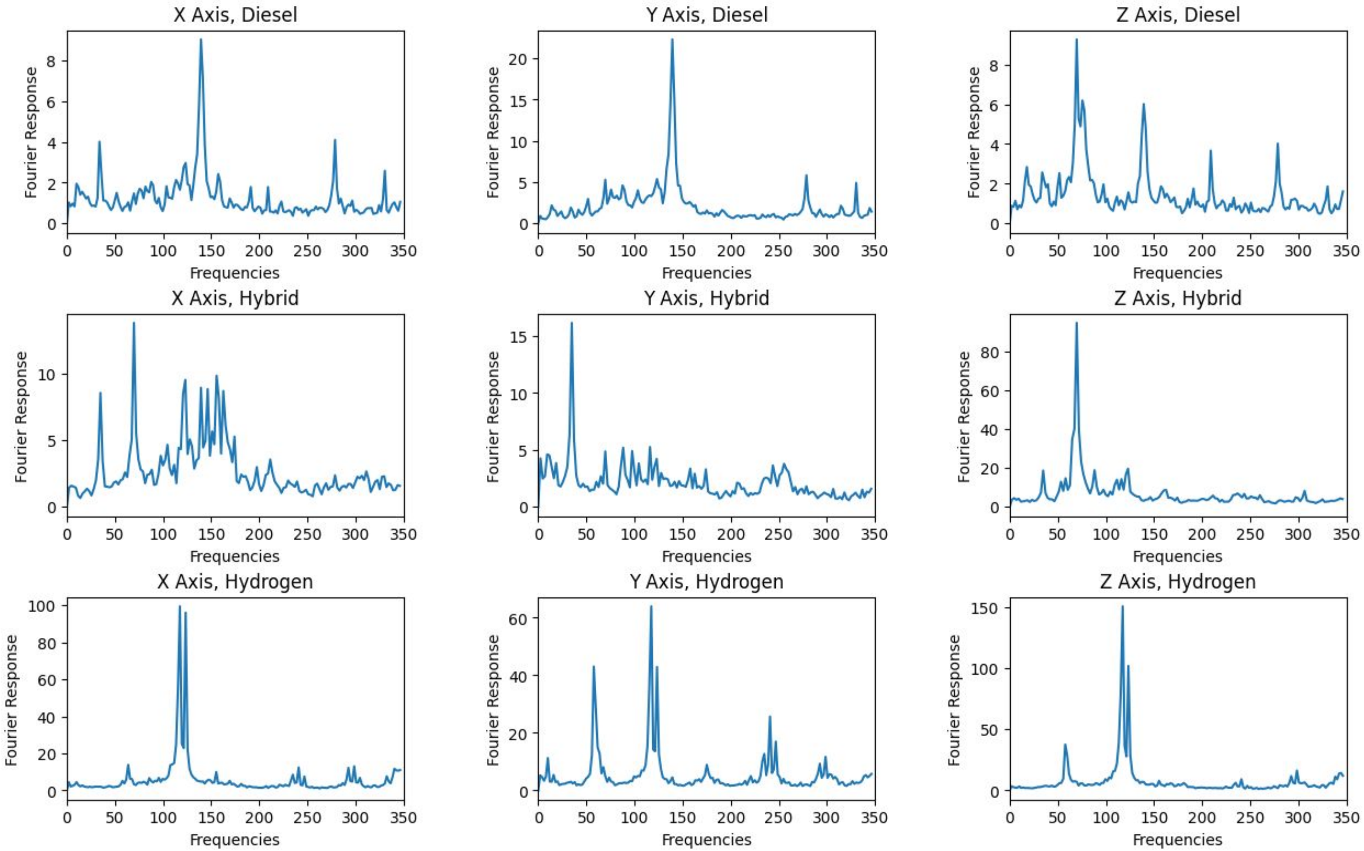
Overall Fourier Transforms Comparison - Bus Types, Moving



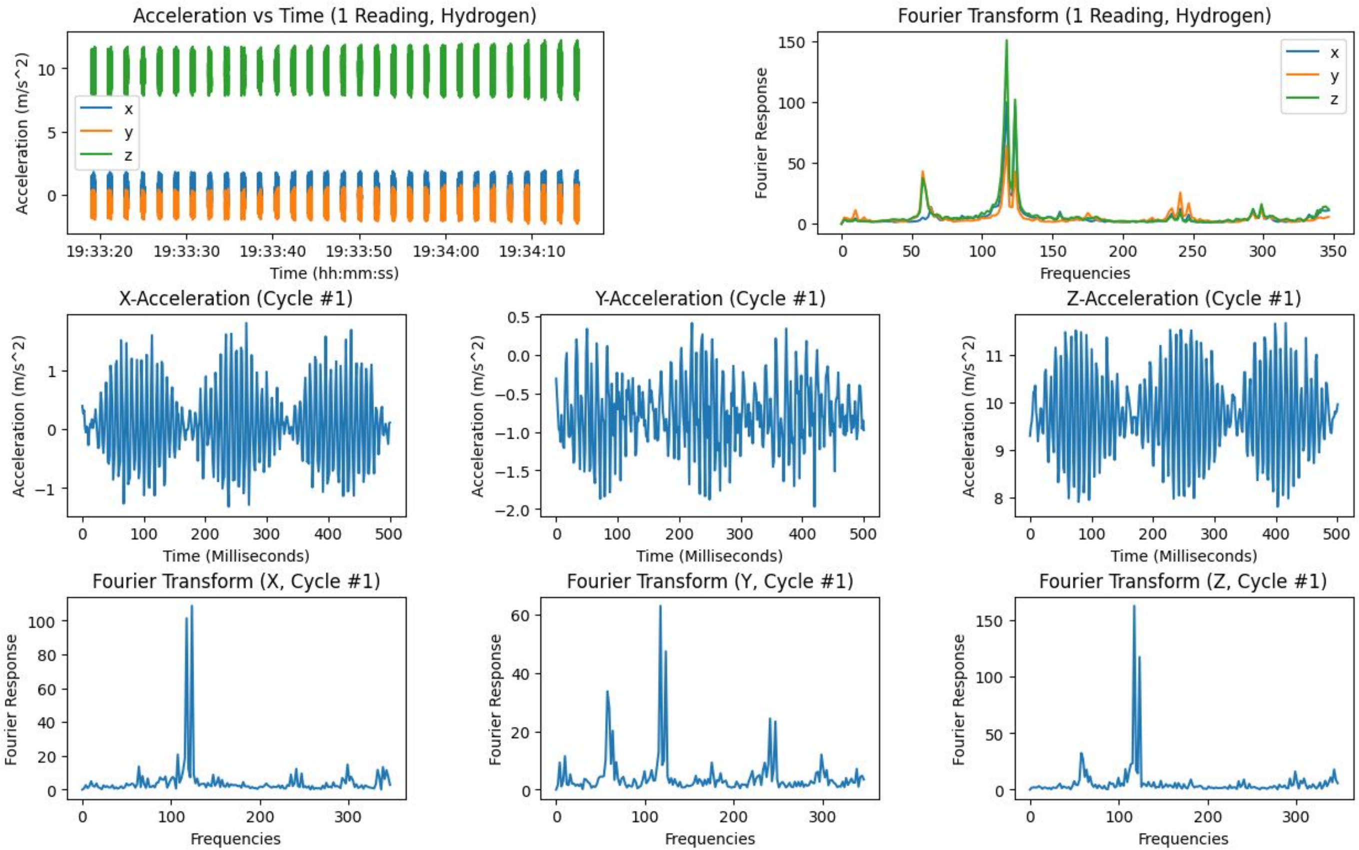
Overall Fourier Transforms Comparison – Bus Types, Idle, Scaled



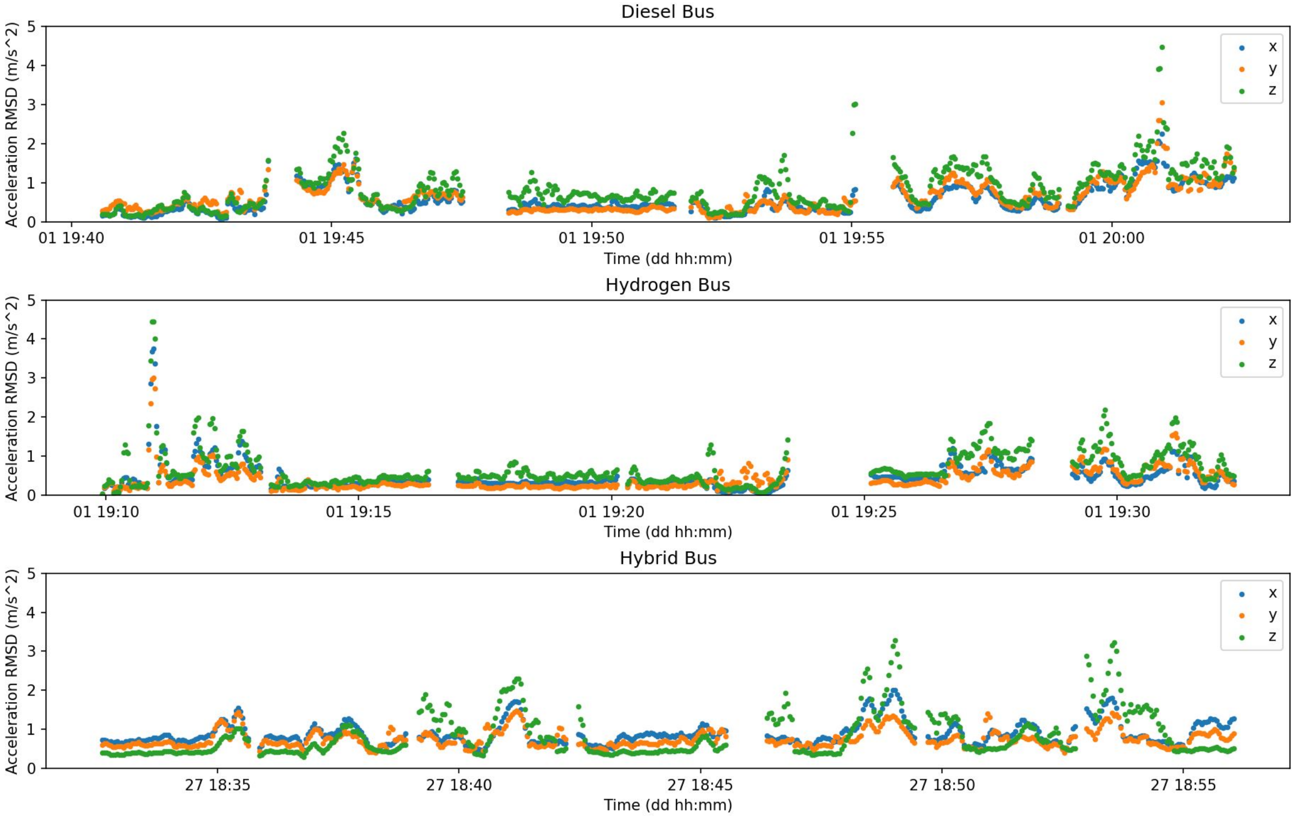
Overall Fourier Transforms Comparison – Bus Types, Idle, Unscaled



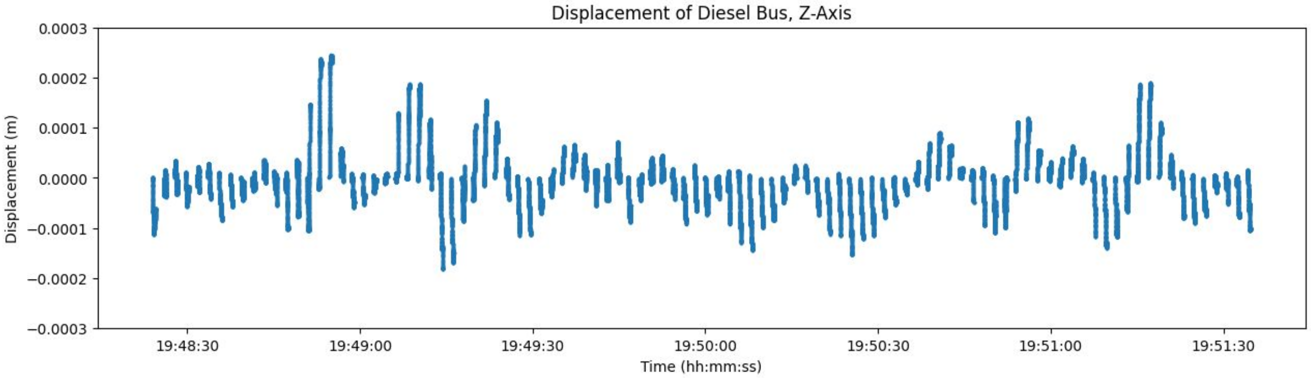
Acceleration-Time Graph and Fourier Response for Idle Hydrogen Bus

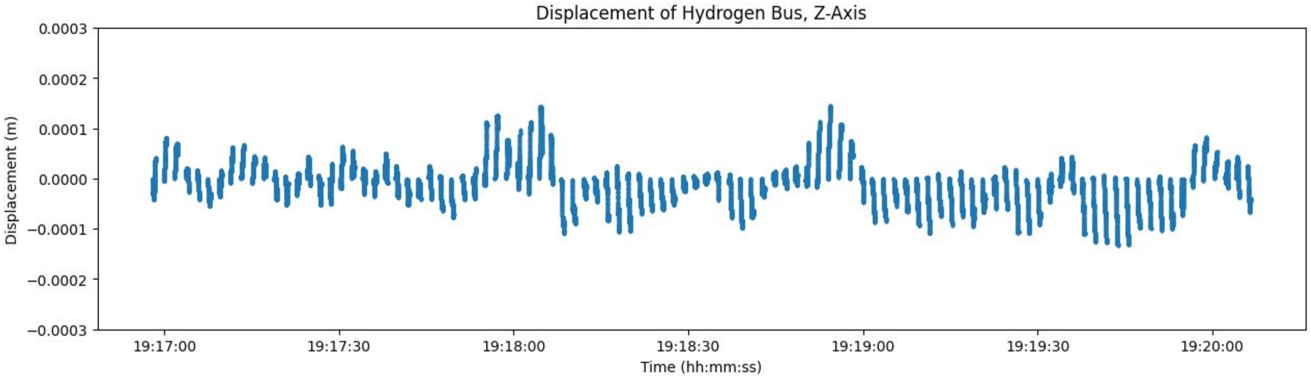


Acceleration RMSD Comparison Across Bus Types

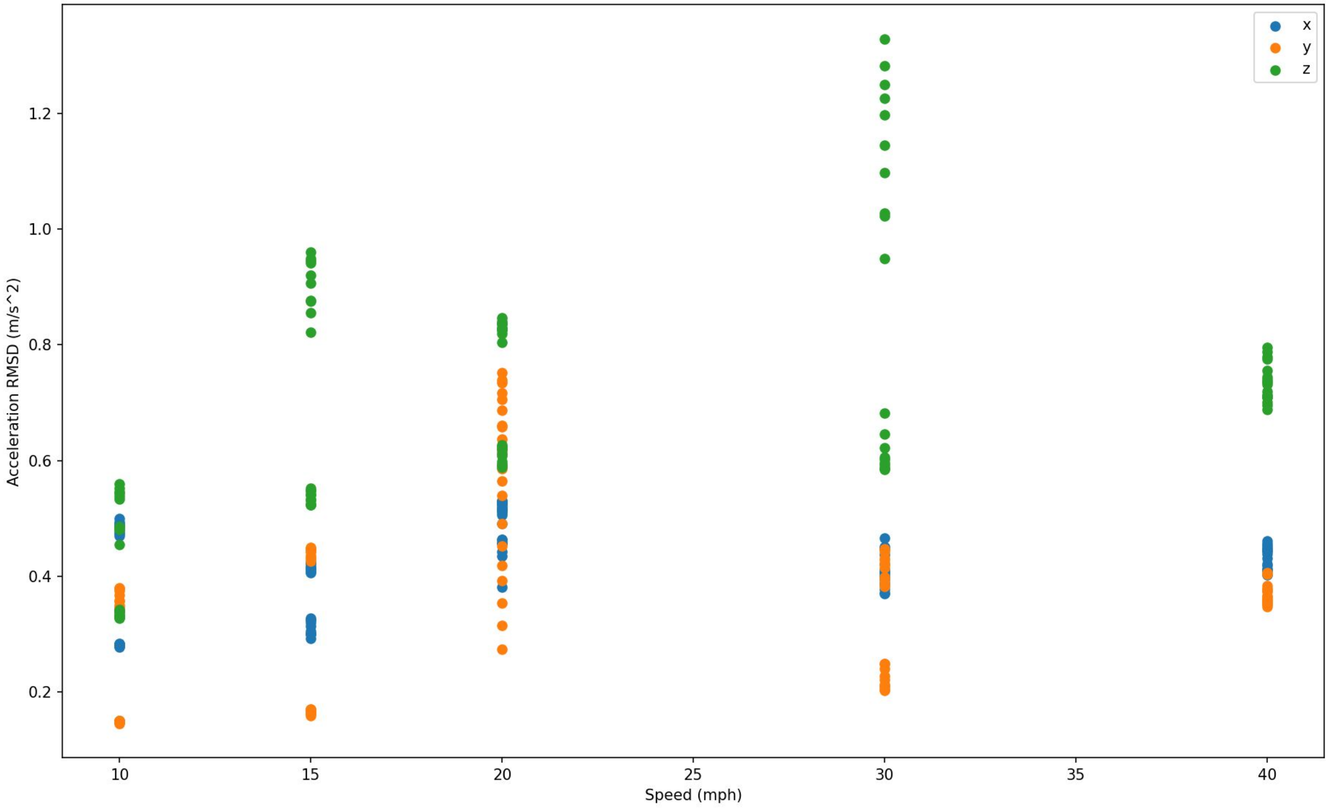


Z-Axis Displacement of Different Bus Types

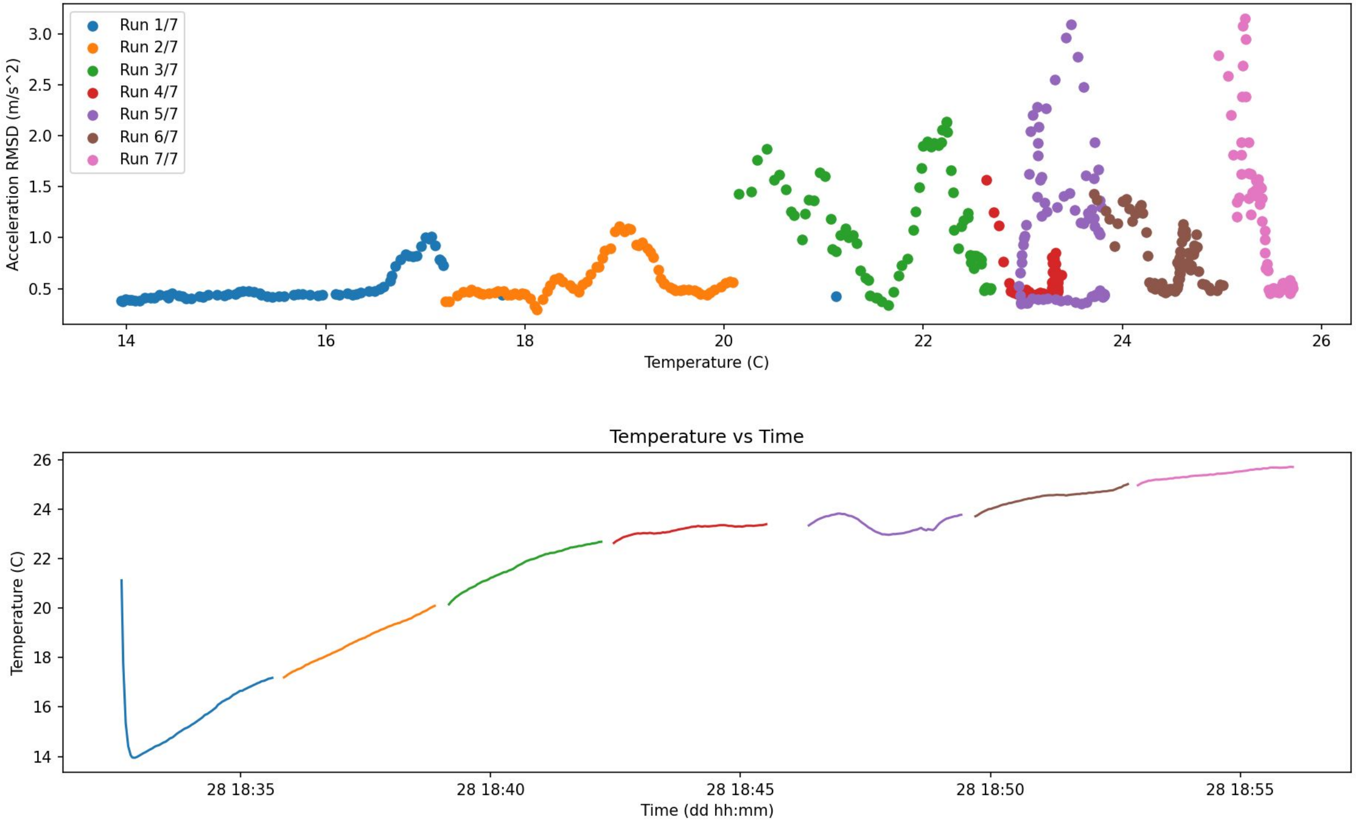




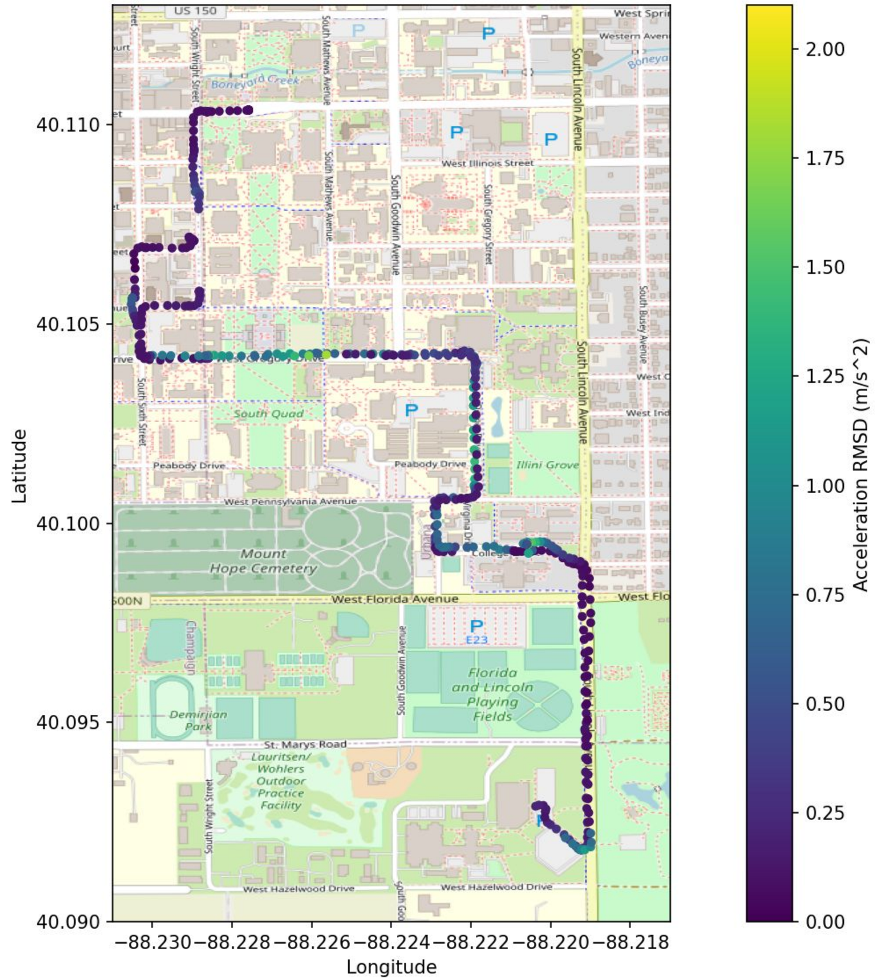
Speed vs Acceleration RMSD



Temperature vs Acceleration RMSD



Acceleration RMSD vs Location



Discussion

Overview – Hybrid Buses

- 1 m/s^2, 120 Hz oscillations in hybrid buses
- Some frequencies around 40 Hz and 70 Hz
- Vibrations fairly uniform across axes

Overview – Bus Type Comparison

- Fourier transform frequency peaks
- Beats in hydrogen bus acceleration
- Hydrogen bus was smoothest ride

Overview – Other Factors

- Speed vs acceleration
- Temperature vs acceleration
- Location vs acceleration

Further Research

- Variable isolation
- Device design improvements
- Effects of vibrations on the human body
- Possible causes of vibrations