

# An Analysis of Anomalous Accelerations Measured on Amtrak Railways

Lauren Gregory, Daniel R. MacLean, Kavya Vaidya, Brian Franklin, Kevin Na

Course Administrator: Professor George Gollin

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

April 26, 2019

## Abstract

The condition of Amtrak railways is investigated in terms of the frequency spectrum and severity of anomalous accelerations caused by faults and imperfections in the track, experienced while aboard intercity trains. Two major Amtrak routes are analyzed: the *Illinois Service* route connecting Chicago and Champaign, Illinois, and the *Hiawatha* route between Chicago and Milwaukee, Wisconsin. The acceleration data are analyzed in both the time-domain and frequency-domain (via Fourier Analysis). The time-derivative of the acceleration (the *jerk*) is also analyzed in order to better characterize the nature of jolts caused by track irregularities. All measurements are carried out with specially designed printed circuit boards consisting of a variety of sensors mounted on breakout boards and sold by Adafruit Industries. The sensors are controlled by an Arduino MEGA 2560 microcontroller. A number of additional test runs are carried out that also yield potentially useful raw data. These tests include rides on the Champaign-Urbana Mass Transit District buses, rides on Chicago Metra Southwest Service and BNSF trains, and even a set of data collected on Italian inter-city trains.

## 1 Introduction

The National Railroad Passenger Corporation (commonly known as Amtrak) provides rail access to the entirety of the contiguous United States, as well as a number of Southern Canadian cities. The long-distance, intercity railroads that constitute Amtrak’s overall transportation network span over 21,400 miles\* of track across 46 states. In recent decades, however, the United States’ infrastructure has seen itself surpassed in terms of both efficiency and quality by many nations around the globe. The most stark examples are those of most European nations and developed Eastern nations such as Japan and China. Many potential reasons underlying this deterioration of infrastructure within the U.S. have been proposed by experts across numerous fields, however most analyses appear to agree that the primary cause for the decline is a lack of government spending and support [6]. According to the American Society of Civil Engineers (ASCE), the general infrastructure of the United States (including passenger railways) is “mostly below standard,” [compared to other developed nations] exhibiting “significant deterioration,” with a “strong risk of failure”[3]. In terms of passenger railroads, it has been determined that a majority of the track that makes up the routes operated by Amtrak has outlived its operational lifetime. That is, most passenger railway track is in

---

\*34,000 kilometers.

need of either extensive maintenance or replacement. Though the United States Government provided Amtrak with a large enough budget each year to keep passenger routes safe for use [5], there is little extra funding that remains for track improvement. This has led rides on the Amtrak railways to become increasingly rough and uncomfortable for intercity riders, garnering a generally negative reputation for passenger rail in the United States as a whole [3][6].

With this analysis, we hope to provide a useful, quantitative assessment of the state of Amtrak railways primarily centered around the Chicagoland area in Illinois. We present this report in the hopes that our model of analysis used to study these tracks may in the future be successfully applied to all Amtrak railways (and perhaps other forms of transportation systems) in order to ultimately improve the quality of the American rail transportation system.

## 2 Hardware

The Data-Acquisition System (DAQ) implemented in our studies of Amtrak railways was designed and constructed based on general designs provided by course administrator Professor George Gollin. The primary hardware element linking the DAQ together is the Arduino MEGA 2560 8-bit microcontroller. The Arduino MEGA 2560 features 256 kilobytes of volatile flash memory in addition to 8 kilobytes of random-access memory (RAM), a built-in 16 channel/10 bit Analog-to-Digital Converter (ADC), a hardware Serial Peripheral Interface (SPI) and is compliant with the Inter-Integrated Circuit communication (I2C) protocol that was implemented in the design of the DAQ. Available for electrical connection are 54 General Purpose Input/Output pins. Using the I2C protocol, we are able to string together a number specific breakout-board sensors and input/output (I/O) devices. All sensors were mounted to breakout-boards and sold by Adafruit Industries. After approximately two months of breadboarding different arrangements of sensors, we arrived at a final list of five primary sensors and I/O elements. These devices are tabulated in Table #1. The LSM9DS1 measures acceleration, gyroscopic motion and magnetic field strength all in three dimensions (for a total of nine degrees of freedom; three Cartesian axes each). All of these measurements are written to their respective columns in the text file. The Adafruit Ultimate GPS breakout board may be configured to read-out data at 1Hz, 5Hz, 10Hz, however 10Hz is not recommended as it may cause readings to become unreliable. The GPS is able to accurately report its position to within roughly 5 meters. The DS3231 Real-Time Clock was used to synchronize the time read-out from the GPS in order to match our position data with the time during any given data collection run. The 5V Ready Micro-SD breakout board is used to write all data received by the Arduino through the I2C communications bus to a single, organized .txt file that is then exported for analysis [5]. The I2C bus has the advantage of only requiring *two* communications lines for full data processing capability; a serial data-line (SDA) and a serial clock-line (SCL) [8].

### 2.1 Adafruit Ultimate GPS Breakout Board

The Adafruit Ultimate GPS presented the largest challenge in terms of measuring and recording useful data. It receives data from satellites at 5 Hz in our version of the code, but that rate can be adjusted. The data it receives comes in two long strings called sentences, and there are two types, \$GPGGA and \$GPRMC [11]. They both contain geographical coordinate information, the date and time, the speed of the device, and the heading. However,

only \$GPGGA sentences contain information regarding the altitude of the device and the number of satellites it is communicating with. We originally started out using Adafruits Ultimate GPS Hardware Serial Parsing example code, but quickly realized that it was not working correctly. When directed with that code, the GPS would only return accurate data if it was the only device being controlled by the Arduino, which was a problem for our experiment which required multiple. We worked as a group with Professor Gollin to find a way to read the GPS in a way that would not interrupt the other devices or inhibit accuracy. He discovered a way to get the GPS to return as much data as it could receive at a given moment, move on to the rest of the loop once it encountered a blank character in its received sentence, and pick up right where it left off. This function, called `gps_query` is defined outside the loop and is called inside of it. We also included writing GPS data to the card inside of an if statement that inserted NaN (“Not a Number”) characters into the file, so the formatting of the tab separated text file would be intact if the device receives no GPS data during a loop or if a \$GPRMC/\$GPGGA sentence is received.

## 2.2 DS3231 Real Time Clock

The DS3231 Real Time Clock is a precision clock used to synchronize the time read-out from the GPS in order to match our position data with the time during any given data collection run. It contains an extremely accurate 32 kHz I2C-Integrated RTC/TCXO/Crystal which is located inside the chip unlike most RTCs which have crystals located externally [7]. External crystals keep time with low current draw however they have slight drift when the temperature changes because temperature changes the oscillation frequency slightly and accumulates overtime; by putting the crystal internally the DS3231 RTC reduces the drift caused by temperature change giving it a more accurate reading. In addition to putting the crystal internally Adafruit put a temperature sensor in order to compensate for the frequency changes by adding or removing clock ticks so that the timekeeping stays on schedule. This RTC is an all in all package which Adafruit topped off by adding a coin cell on the back of the sensor allowing the user to take years of precision timekeeping even if the main power is lost [7].

## 2.3 LSM9DS1 Nine-Axis Accelerometer, Gyroscope, and Magnetometer

As stated previously, the LSM9DS1 is a nine degree-of-freedom inertial unit that measures acceleration, gyroscopic motion and magnetic field strength in all three dimensions. Each units chip contains 3 sensors; a 3-axis accelerometer, a 3-axis magnetometer, and a 3-axis gyroscope. The first sensor, the 3-axis accelerometer, measures gravity in order to tell the user which direction is facing down towards the Earth in addition to informing the user how fast the board is accelerating in 3D space. The accelerometer on the LSM9DS1 has  $\pm 2/\pm 4/\pm 8/\pm 16$  [g] (no  $\pm 6$  [g] range unlike its predecessor) [9]. The second sensor is a 3-axis magnetometer that detects which direction the magnetic north lies by sensing where the strongest magnetic force is coming from. The LSM9DS1 magnetometer has  $\pm 4/\pm 8/\pm 12/\pm 16$  Gauss ranges. The third and final sensor is a 3-axis gyroscope that uses Earths gravity to measure spin and twist and ultimately help determine orientation. The gyroscope has degrees-per-second ranges of  $\pm 245/\pm 500/\pm 2000$ . Finally, they equipped the LSM9DS1 inertial unit with both I2C and SPI interfaces making it easy to connect it to the Arduino using only 4 wires; the serial data-line (SDA) and the serial clock-line (SCL) for connection tot he I2C bus as well as Vin (5 Volts) and ground (GND) for power [9].

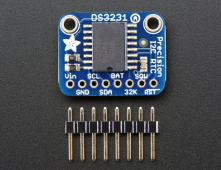
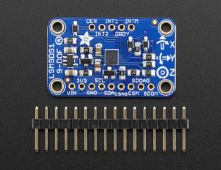
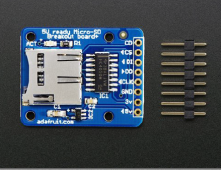
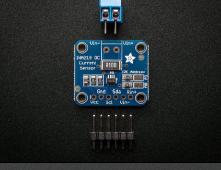
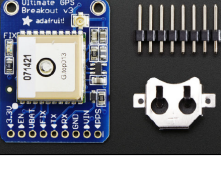
| Image  | Sensor                   | Description                                  |
|--|--------------------------|--|
|   | DS3231 RTC               | Extremely accurate real-time clock.          |
|   | LSM9DS1                  | 9-Axis Accelerometer.                        |
|   | 5V Ready Micro-SD        | Micro-SD card writer; outputs data from PCB. |
|   | INA219                   | Current/voltage/power sensor.                |
|  | Ultimate GPS Breakout v3 | Global Positioning Satellite down-link.      |

Table 1: Summary of hardware used for the DAQ. All images of sensors retrieved from respective datasheets published by Adafruit Industries and listed in the **References** Section.

## 2.4 INA219 Current/Voltage Sensor

The INA219 is a high side DC current sensor which is used for power-monitoring related problems. It reports its measurements, through the I2C communication protocol, of both the high side voltage and DC current draw with 1% precision. Most current-measuring devices are only good for low side measuring. Low side measuring is when you stick the measurement resistor between the target ground and true ground. This can prove troublesome as most electronic circuits behave abnormally when the ground references change and move with varying current draw [6]. The INA219 chip can handle high side current measuring, up to +26VDC, even though it is powered with only 3 or 5V. Adafruit even integrated a function that will report back the high side voltage to the user making it great for tracking battery life. This allows us to monitor which sensors are using the most power during certain situations. It also hosts a precision amplifier that measures the voltage across the 0.1 Ohm resistor allowing it to measure up to  $\pm 3.2$  Amps with resolution of 0.8 mA as well as a max current of  $\pm 400$  milliamps and corresponding resolution of 0.1 milliamps. Adafruit completed this breakout unit by using I2C communication protocol making it as easy as connecting 6 pins to get the unit ready for immediate use [6].



### 3 Design of the Data-Acquisition System

The first challenge of designing the breadboarded circuits was to figure out what sort of data we were aiming to collect while on the train, and then the instruments we wanted the Arduino to connect to that would correspond with the respective data. Velocity and acceleration are obviously necessary as we are measuring anomalies in the tracks that cause unexpected kinematic “jolts” felt aboard the train. GPS data was also necessary, in order for us to see which sections of the track were associated with the measured jolts and lurches. After that, we would need a way to communicate with the Arduino and also be able to tell what it was doing at the moment. We needed a way to store the data as well, so we could analyze it later. Lastly, we required a way to verify that power was being supplied to the Arduino and if it was constant, i.e. no sudden loss of power. So respectively, the components we had chosen for our breakout boards were the accelerometer, the GPS, the keypad and LCD screen, the Micro-SD card chip, and the INA219 current/voltage/power sensor. These chips then were attached to the circuit, and the power source would be our computer for the breakout boards. A photograph of one of the five breadboard circuits is displayed in Figure #1. For the PCB we would need another source of power since the devices would have to act as standalone systems that could collect data in the field without a laptop. After determining which sensor chips we required in our boards, it was simply a matter of following the schematic(s) and assembling the final version of device. After designing the data acquisition system breadboard circuit, we did trial runs on the bus as mentioned previously and began to assemble the official data acquisition system on the PCB. The final PCB version of our device also includes a battery pack that holds five 1.5 Volt AA batteries. This eliminates the need to plug the DAQ into a laptop for power.

Through application of the EAGLE Schematic generating software, the breadboarded design was translated into a printed circuit board (PCB) by editing Professor Gollin’s original design for the *general* schematic of the device (which contained all sensors available from Adafruit). Once soldered, the final PCBs served as our primary data acquisition system (DAQ) for the remainder of our semester-long investigations into Amtrak rail anomalies. A photograph of one of the completed printed circuit boards is displayed in Figure #2.

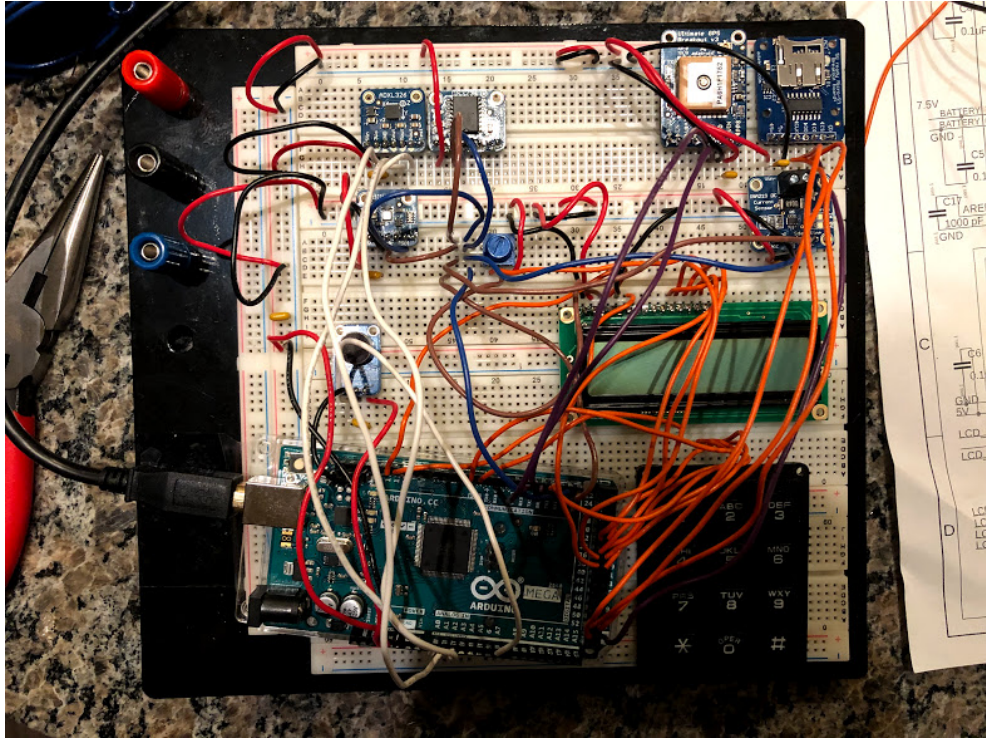


Figure 1: Image of the breadboarded DAQ while still in the design phase.

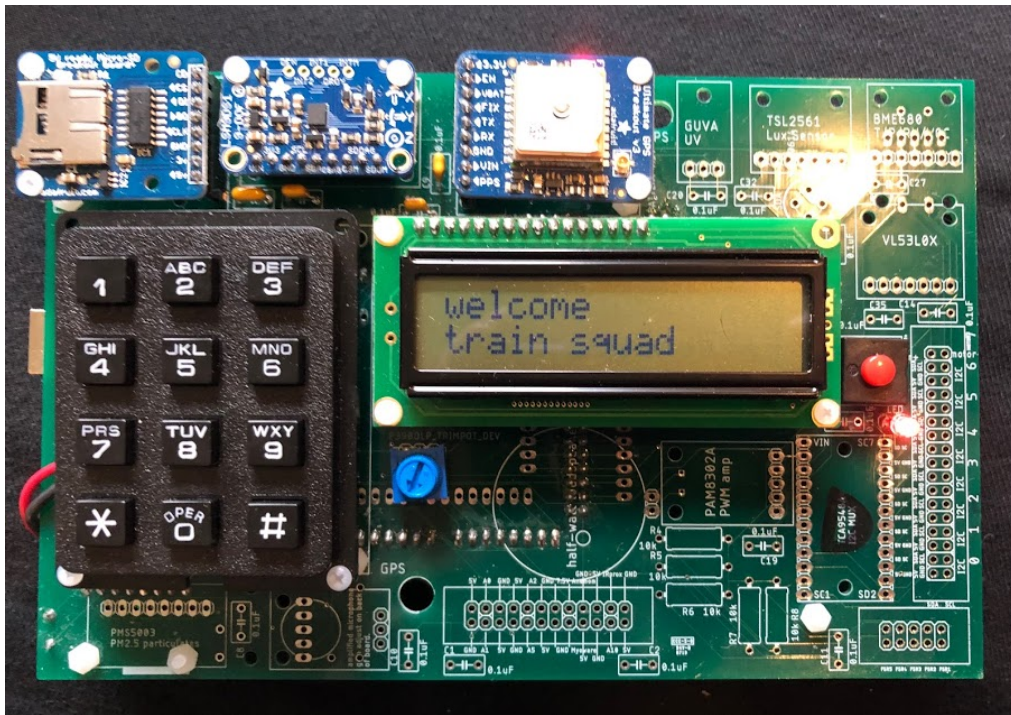


Figure 2: Image of the final, soldered PCB version of the DAQ.

## 4 Software

### 4.1 Arduino Code

The data acquisition software was created by programming an Arduino Mega 2560 to record measurements taken by the various Adafruit breakout-boards to a Micro-SD card. The code run by the Arduino microcontroller was written in the Arduino Integrated Development Environment and uses a C/C++ compiler. The final program used to acquire the data discussed in this paper was the sixth version. The code begins by importing libraries, initializing global variables and establishing links to the devices we are using. The setup function powers-on all devices and checks that they are working properly, and displays error messages on the serial monitor and LCD screen if one does not start up. It also displays a welcome message to the LCD so we can be sure everything has started up correctly. Then the loop begins.

We have a loop counter global variable to keep track of how many times the loop has run. If the loop counter is equal to 1, the Arduino is instructed to open a file on the Micro-SD card. We kept a consistent numeric naming convention for our files, so the code automatically checks if the file `DATA000.TXT` exists on the card. If it does, it keeps adding 1 to the three digits at the end until it finds a unique filename, and opens a file in that name. We found this feature to be useful because we did not have to keep uploading a new version of the code with a different filename written in every time we took more data. In addition, the files in numerical order would also correspond to data taken in chronological order, which helped us keep track of our data. Once the file opens, a header line is written. We decided on a tab separated text file for raw data output that has 22 columns. Then, at a frequency of about 1000 Hz, acceleration data is written to the text file.

Our data acquisition code utilizes the keypad at the end of the loop function for a variety of responsibilities. Every loop, the code checks if a key has been pressed. If it has not, the microcontroller continues onto the next loop. If it has, a series of if statements allow a certain function to occur based on what key was pressed. Most important is closing the text file, dictated by the 1 key. Writing to an SD card with an Arduino, a file has to be both opened and closed in order for it to be saved to the card. If a file is opened and not closed, it will show up on the SD card as an empty file. Pressing 1 on our keypad closes the file and exits the loop, effectively ending data acquisition. If the Arduino loses power and the file is never closed, all of the data will be lost. In an early version of our code, we had the file opening and closing every time through the loop. This was inefficient because that operation takes time, and we wanted our acceleration data to be sampling at as high of a frequency as possible. Pressing 2 on the keypad displays the filename on the LCD screen. This was useful to keep track of which files corresponded to which trips. Keys 3-7 correspond to different measurements regarding the circuit board taken by the INA219. Key 8 writes a 1 to the event column in the file to signify entering a stop, and key 9 writes a 2 to signify leaving a stop. We used this to make sure what certain spikes or lulls in acceleration could correspond to when looking at the data later. In the same column, the '\*' key writes a 3 to signify an event occurring. This was to look at spikes in data originally, but we later discovered that when using multiple boards to acquire data, simultaneously pressing these keys and recording an event allowed us to sync up the data and display it better. When a key that is not 8, 9, or '\*' is pressed, or no key is pressed at all, a 0 is written to this column. The 0 key prints the current "millis" value (that is, the number

of milliseconds since the DAQ has been activated) to the LCD. This is helpful when recording events, because when knowing what time it is according to the Arduino, we can keep track of what event happened at what time to make analyzing big spikes in our data easier. Finally, the '#' key took a long time to decide on a purpose for, but we decided to include some entertainment for the train rides and program it into a fortune teller. We call it the magic conch, and when the '#' key is pressed, it displays a different message on the LCD depending on the last digit of the current `millis` value. This ends the loop. In addition to `gps_query`, another function is defined outside the loop. `LCD_message` takes in two strings, clears the LCD screen, prints the first one on the first line of the LCD screen and the second on the second line. This is useful because if we just want to display strings on the screen, this function streamlines the process of clearing the screen and placing the cursor.

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| * | 0 | # |

Table 2: Schematic of 4x3 keypad layout.

The keypad-command layout (as programmed in the Arduino IDE and shown schematically in Table #2) for the final version of the DAQ is listed below:

- 1: Close file, end program. This is necessary in order to save any data for a given run.
- 2: Display filename on LCD
- 3: Display shunt voltage on LCD
- 4: Display bus voltage on LCD
- 5: Display current on LCD
- 6: Display power on LCD
- 7: Display load voltage on LCD
- 8: Record in-file that the train has *entered* a stop.
- \*: Record in-file that the train has *exited* a stop.
- 0: Display current milliseconds value (since the DAQ was activated)
- #: Fortune-Teller

## 4.2 Python Code

We used Python for the mainstay of data processing due to our familiarity with it and due to the ease with which standard libraries such as Numpy handle large 2D and 3D arrays of data. Our program imports the text file directly from the Micro-SD card and writes a new `.csv` file of processed data. We use the Python libraries Numpy and Pandas in this code to these raw data recorded by the DAQ. Numpy is a library of functions built for manipulation

of multidimensional arrays of data. Pandas is a similar library, designed specifically for analysis of large datasets and straightforward visualization applications. The Python program begins by establishing input and output filenames. When processing a new file, this is the only part that will need to be edited to reflect the different files. Using Pandas, we turn each of the columns from the tab separated file into a list object. We then converted each list to a Numpy array. We did this in order to avoid using a `for` loop to fill an array on such a large set of data, which would prove inefficient and time-consuming. As mentioned above, the event column is useful to synchronize data from multiple boards. After converting everything to Numpy arrays, we find the index of the first event recorded from us pressing the '\*' key to signify the beginning of data collection. We find the `millis` value corresponding to that index, then subtract that value from the entire array, setting the beginning index `millis` value to 0. When the arrays are then put into a `.csv` at the end of the script, only the beginning index through the end of the array is added. Therefore, the resulting `.csv` has `millis` start from 0, when the event key was pressed, from every board no matter when the device was turned on.

The GPS returns latitude and longitude values in a non-conventional format, `DDMM.MMMM` where the `MM.MMMM` indicates minutes of arc expressed as a *decimal* instead of in terms of seconds of arc. In order to convert this to just degrees for mapping purposes, we have a function called `degrees` that takes in a coordinate. This coordinate modulus 100 gives us the minutes value, and the coordinate floor divided by 100 gives us the degree. Adding the degree to the minutes divided by 60 gives us the final value solely in degrees, which is what the function returns. Using this function, we convert latitude and longitude raw output from the GPS to degree values. Since the GPS samples at a slower rate than the accelerometer, the data columns that come from the GPS have a significant amount of `NaN` characters in them. We then strip the data from the GPS to only non-`NaN` values. (ex: `longitude_stripped = longitude[np.isnan(longitude) == False]`) Then, to find the indices of the entire data set where GPS data appears, we look through the entire array and return indices where the values are *not* written to file as `NaN` (ex: `lon_index = np.array([index for index,value in enumerate(longitude_raw) if np.isnan(value) == False])`). These indices are used to create a new `millis` array that only includes `millis` values that correspond to the GPS data points. Then, using these stripped data values, each array of GPS sourced data is interpolated over the full `millis` array. We made sure to check the accuracy of a simple Numpy linear interpolation, and it filled in the blanks of GPS data very well and matched maps made solely from the raw data. Finally, using Pandas again, we convert all of our data to a `DataFrame` and then output it to a `.csv` file under the filename determined in the beginning. Now we have processed data ready to display. A listing of each measurement made by the DAQ that is output as data and written to the Micro-SD card breakout board for further use is given in Table #3.

### 4.3 Mapping with Tableau Desktop

Due to the importance of geographical coordinates to our analysis, we wanted to find a way to overlay our results on a map. We used Tableau Desktop for its mapping capabilities, emphasis on data visualization, and the ease in which it converts file columns to arrays that can be plotted. It was useful for runs made with multiple boards because it can create unions between multiple files and plot multiple boards data on one plot with the ability to differentiate

| Data Column Name | Quantity/Data-Type                                |
|------------------|---|
| loop count       | # of times through loop                           |
| millis           | Milliseconds-index [ms]                           |
| Gx               | Gyroscope $\hat{x}$ reading [deg/s]               |
| Gy               | Gyroscope $\hat{y}$ reading [deg/s]               |
| Gz               | Gyroscope $\hat{z}$ reading [deg/s]               |
| Ax               | Accelerometer $\hat{x}$ reading [g]               |
| Ay               | Accelerometer $\hat{y}$ reading [g]               |
| Az               | Accelerometer $\hat{z}$ reading [g]               |
| Mx               | Magnetometer $\hat{x}$ reading [Gauss]            |
| My               | Magnetometer $\hat{y}$ reading [Gauss]            |
| Mz               | Magnetometer $\hat{z}$ reading [Gauss]            |
| pitch            | Rotations about $\hat{y}$ -axis [degrees]         |
| roll             | Rotations about $\hat{x}$ -axis [degrees]         |
| heading          | Rotations about $\hat{z}$ -axis [degrees]         |
| time             | Current time from GPS [hh:mm:ss.ms]               |
| latitude         | Latitude [deg:min.min]                            |
| longitude        | Longitude [deg:min.min]                           |
| speed            | Measured by GPS [knots]                           |
| direction        | Measured by GPS [radians]                         |
| satellites       | Number of satellites currently fixed to GPS       |
| altitude         | Distance above sea-level measured by GPS [meters] |
| events           | User-defined markers                              |

Table 3: Listing of all raw-datasets output by the Data-Acquisition System as column-vectors in a `.txt` file.

between the sources. In addition, it allows different data parameters to be used as plot details such as color and size. This is useful for analysis of multiple datasets at the same time. The primary maps made with Tableau consist of what we shall refer to as 'jolt-density' maps, highlighting regions in which high-amplitude jolts of acceleration were experienced aboard a train.

#### 4.4 Additional Analysis with OriginLab

In addition to Python, OriginLab was used to analyze the arrays of data produced by the DAQ. Due to its ability to quickly and easily generate 2D, multi-axis/layered plots, we found that this program was significantly more efficient than Python. Origin also supports a built-in Fast-Fourier Transform function (FFT) that was useful in the analysis of unexpected high-frequency, low-amplitude bursts of acceleration discovered in the LSM9DS1 data for both the MTD test-runs and Amtrak runs.

## 5 Data Collection

The process of collecting data during rides on both trains and buses consisted of one or more members of our group activating then monitoring the DAQ throughout the course of the ride. Once activated, the DAQ device is designed to continuously take data without any additional input from the user, with the exception of the `entering-stop` and `exiting-stop` as well as the `event` marker, which must be entered manually via the keypad. These markers appear in the raw data within a specific "event" column. The DAQ was positioned such that the  $\hat{x}$ -direction was oriented *forwards*, towards the front of the train or bus<sup>§</sup>. The  $\hat{y}$ -axis then pointed side-to-side with the right/starboard side representing the *positive*  $\hat{y}$ -direction. Finally, the  $\hat{z}$ -direction corresponded to the vertical (up/down) direction. The  $\hat{z}$ -axis constantly experiences a downward acceleration due to the gravity of the Earth, *in addition* to any accelerations felt by the DAQ. It is important to ensure that the DAQ remains in this configuration throughout the course of the ride, as well as to adequately mark all major "events" that may produce confounding results in the raw data obtained for that run. Events such as these include but are not limited to:

- Entering and exiting stops (there are specific keypad commands for these particular events)
- Major jolts/bursts of acceleration experienced during the ride that warrant specific analysis
- Accidental perturbation of the DAQ (i.e. bumping it; this will produce a large spike in the acceleration profile)
- If the INA291 readings indicate that the power is running low and the batteries may die

The keypad on the PCB was configured for easy use by the operator (see: keypad command list in Section #4). MTD buses we chose to ride covered routes that we had ridden before, ensuring that we were all at least fairly familiar with them. However, since multiple boards began to show anomalies in the position data, we began to look for other sources of error that could have been causing it. These anomalies in the data will be discussed in a later section of the report, since it has affected our findings significantly. One thing we did to attempt to pinpoint the

---

<sup>§</sup>There are a handful of datasets from runs in which this coordinate-axis was altered, for instance, when springs were used to suspend the DAQ from the handrail of an MTD bus. These rearrangements were accounted for in the analysis of the raw data.

source of error was to have all our boards on at the same time for the same trip, and figure out if the errors in our GPS data was board specific, location specific, or time specific. All of our boards had the same data points that caused error at the same locations and very close in timing. After figuring out how to use the data we received and making sure our boards worked, we attempted to collect data with the PCB boards. This was much simpler since they were able to be held/positioned in a fixed location very easily, and did not require connection to an external power source. While simultaneously dealing with the issues we were facing with the GPS at the time, we began to test other ways to reduce the constant micro-vibrations that were being shown in the MTD data. One method that has been proposed was attaching the springs to both sides of board and then attaching it to the upper railing of the bus. This would act as some sort of suspension, and the vibrations would then come in a sinusoidal wave which could then be Fourier transformed and removed from the data. However, the springs rotated a fair amount in the  $\hat{x} - \hat{y}$  plane, which was parallel to the surface of the Earth, during turns of the bus. We analyzed the data and found that the springs did dampen the effect of the constant vibrations that were received from the bus itself, so we moved forward with the idea of using springs. Since we were measuring the large shifts in velocity, i.e. the bumps and lurches in the tracks, small micro-vibrations were necessary to remove from the final datasets that would be acquired during Amtrak rides.

In Appendix II, a handful of photographs taken during both test-runs and Amtrak trips are included. These have been included both for posterity and to serve as illustrative examples of the process of data collections as it was carried-out.

## 6 Data and Analysis of Results

The data were analyzed using the suite of software applications mentioned in section 4. The raw-data written to the micro-SD chips for each PCB were exported as `.txt` files into Python which was used to process the data to prepare it for analysis. All measures of acceleration *all accelerations* are presented in units of the Earth’s gravitational field<sup>†</sup> or “g”, unless otherwise specified. Though the LSM9DS1 Accelerometer measures accelerations for all three Cartesian axes, we occasionally calculate and use the *net*-acceleration felt by the DAQ at any given time. We calculate the net-acceleration by simply adding each of the measured  $\{\hat{x}, \hat{y}, \hat{z}\}$  components in quadrature:

$$a_{\text{net}} = \sqrt{a_{\hat{x}}^2 + a_{\hat{y}}^2 + a_{\hat{z}}^2} \quad (1)$$

It is important to note that, in order to avoid dilution of the  $a_{\hat{x}}$  &  $a_{\hat{y}}$  components of acceleration, the effectively constant acceleration due to the Earth’s gravitational field was *removed* from the  $a_{\hat{z}}$  component before calculation of the *net* acceleration using Equation #1. The diluting effect of Earth’s gravitational field is discussed further in Appendix I. In addition, we frequently make use of the *root mean-square* (quadratic mean) of the net-acceleration to characterize the overall “roughness” of a given ride. When we refer to an *average* of the net-acceleration profile

---

<sup>†</sup>The nominal gravitational field of the Earth is a quantity defined as  $1 [g] = 9.80665 [\text{m/s}^2]$  according to the National Institute of Standards and Technology (NIST).



$a_{\text{net}}$  with  $N$  total data points, we are in fact referring to calculations of the root mean-square given by:

$$a_{\text{rms}} = \sqrt{\langle a_{\text{net}}^2 \rangle} = \sqrt{\frac{1}{N} \sum_i^N (a_{\hat{x},i}^2 + a_{\hat{y},i}^2 + a_{\hat{z},i}^2)} \quad (2)$$

In order to analyze the frequency-spectrum of periodic systems, it is often extremely illuminating to apply Fourier Analysis. This mathematical tool allows us to easily investigate the *frequency-domain* components of a system rather than the time-domain behavior (which is usually readily measured in the setting of a laboratory, or in our case, aboard a train). We apply *Fourier Transforms* to the time-domain net-acceleration profiles measured aboard all trains and buses, in order to identify any prominent, periodic signals indicative of sources of anomalous acceleration that we are interested in analyzing. For a time-domain function  $\Psi(t)$ , the corresponding frequency-domain function  $\Phi(\omega)$  is related through a Fourier Transform by:

$$\Phi(\omega) = \int_{-\infty}^{+\infty} \Psi(t) \exp\{-i\omega t\} dt \quad (3)$$

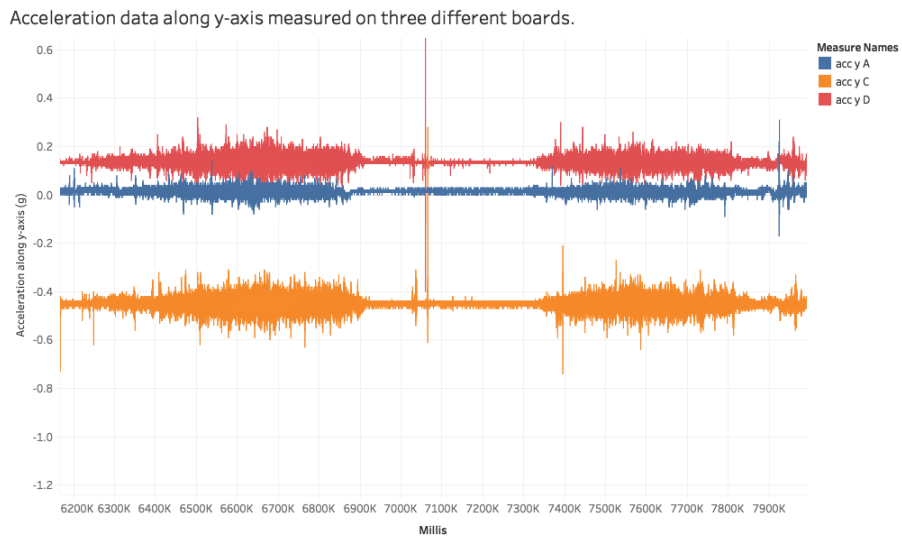
$$\Psi(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \Phi(\omega) \exp\{i\omega t\} d\omega \quad (4)$$

Using OriginLab, we may easily perform Fourier Transforms by using the **Fast Fourier Transform** (FFT) function located under **{Analysis → Signal Processing → FFT}**. This function takes time-domain input data and outputs a variety of useful frequency-domain parameters and graphical representations. Fast Fourier Transforms, however, are *discrete* summations rather than continuous integrals. The units of the *amplitudes* calculated by the FFT algorithms should therefore be in the original units of net-acceleration, [g].

## 6.1 DAQ Synchronization

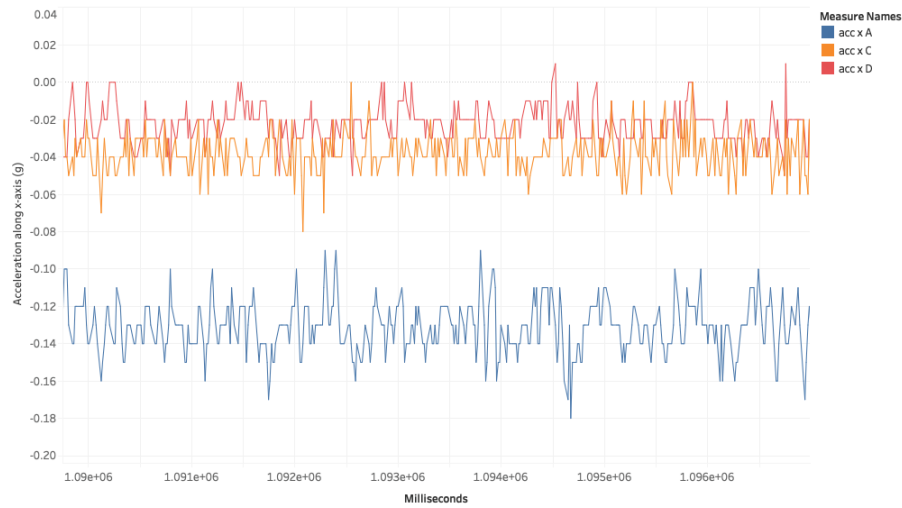
It is obviously of high importance to have each board synchronized in time for a data collection run in which multiple boards are present. It would otherwise be extremely difficult to obtain confirmation that a jolt registered by one board is a real track anomaly. If all four DAQ devices record a jolt at essentially the same point in time however, it is much more clear that the anomalous acceleration was caused by a legitimate fault or abnormality in the track. It was therefore necessary to align the data collected by all boards as closely in time as possible. In Figure #3, two images of the same dataset are displayed, one enlarged to show the level of synchronization between boards. Each color represents the measurements recorded by a single board on the same Amtrak railway.

The acceleration data aligns pretty well across multiple sensors placed on different boards. Major shapes in the acceleration vs time graphs are consistent and discernible throughout the different data sets. The individual acceleration peaks from data of various boards could be aligned to a difference of about 1000 milliseconds (one second). The plots seen in Figure #3 appear to be shifted in the amplitude of acceleration, which is consistent across all three-axes for different datasets. This could be attributed to devices being placed at slightly different angles when they were placed on their respective surfaces. The amplitudes of acceleration for all three boards from their



Millis vs. acc y A, acc y C and acc y D. Color shows details about acc y A, acc y C and acc y D.

Acceleration data along x-axis measured on three different boards.



Millis vs. Avg. acc x A, Avg. acc x C and Avg. acc x D. Color shows details about Avg. acc x A, Avg. acc x C and Avg. acc x D.

Figure 3: Sample net-acceleration vs. time dataset showing the precision to which we are able to align a number of DAQ devices in time. The profile to the right is an enlarged segment of the profile to the left.

respective average values in 'normal' stretches of track (i.e. without large jolts) is about 0.03 [g]. The overall bursts of acceleration along with major jolts align across the data from various boards.

## 6.2 Repairing Corrupted GPS Data

Throughout the course of our investigations into the condition of Amtrak railways, we experienced two consistent patterns of unexpected behavior in the datasets produced by the Adafruit Ultimate GPS breakout-board. We isolated both patterns as begin distinct from each other and classified them accordingly as a *large-scale* anomaly and a *small-scale* anomaly. The large-scale anomaly took the form of a wandering strings of GPS points across an extremely wide range of latitude & longitude values, veering widely off-course from the actual motion of the PCB. Eventually, with the help of Professor George Gollin, we isolated the issue and successfully rectified it. Professor Gollin created

a Python script to repair corrupted GPS data. The anomalies we experienced with our data collection presented an obstacle to getting accurate data, and we knew we had to find a way to fix it. Luckily, while looking at the difference between raw GPS sentences and the parsing code we had been using with Professor Gollin, we noticed discrepancies. When converting a string to an integer, our Arduino was clipping leading zeros off of our latitude and longitude minute values. This meant that, for instance, a point that was actually 4006.0017 would show up in our data as 4006.17. When working in GPS coordinates, that's a big difference. Luckily, since all the output is supposed to have a certain amount of digits, we could look back through our raw data and repair the inaccurate points that were missing digits. That's what this repair script is for.

To begin, the script (called `GPS_repair.py`) imports the raw data file written directly to our DAQ's SD card. It reads it in as a `.csv` with a tab delimiter, and with a for loop reads through the entire file row by row, looking only at the row indices in which navigation data is written to. Since this is our raw data and we have a great number of NaN values in the navigation data columns originally, it checks for that first. If GPS data is there, it then takes a closer look. Using the latitude or longitude value, the repair script pinpoints the index of the period in the string of the GPS value. After that, it counts the number of digits following the period in the value. It is supposed to have 4 digits for a complete value. Throughout the loop, we keep track of how many of number of digits we missed. So there's a counter for 1 missed zero, 2 missed zeros, three and four. When one of these is found through application of a series of `if` and `elif` logic gates, 1 is added to their counter. A new string is then created with the appropriate number of zeros inserted at the correct index, and this is saved in place of the old value in its row. At the end of the loop, below these if statements, the new row is written to the new output file. After this loop, the counter variables are printed out to make sure it is working properly, and both files (input and output) are closed.

### 6.3 Analysis of Jolts Caused by Track Anomalies

In this section we carry-out a quantitative analysis of what we shall interchangeably refer to as *lurches* or *jolts* that occur when an Amtrak train (or Metra train, Italian train, MTD bus, etc.) encounters a track anomaly. These jolts are the rough, short-period bursts a passenger feels when a vehicle traverses rough surfaces or misaligned segments of track. To obtain a more complete picture of the severity of a given jolt/lurch measured aboard an Amtrak train, we analyze the time-derivative of the acceleration (i.e. the third derivative of the position) often referred to as the *jerk* and which we shall represent as  $\mathcal{J}$ . A study carried out by Powell & Palacín in 2015 [1] suggests that the jerk is equally important in the analysis of uncomfortable and potentially dangerous jolts experienced aboard both long-distance passenger trains and commuter railways. If  $a$  is the magnitude of the acceleration at a given time  $t$  and  $r \equiv \sqrt{x^2 + y^2 + z^2}$  represents the position, the jerk may be expressed as:

$$\mathcal{J} = \frac{d\vec{a}}{dt} = \frac{d^3\vec{r}}{dt^3} \quad (5)$$

The jerk is clearly a vector-quantity like acceleration, thus we shall analyze it in terms of the same three Cartesian components we have thus far considered. We calculate the jerk based on an averaging algorithm implemented in a Python script written by Professor George Gollin (see: Section #6.4). In addition to the jerk, we calculate the *displacement* of the train from its nominal position on the rails. This is accomplished with the same averaging

algorithm used to calculate the jerk for a given ride and is discussed in detail in Section #6.4. Analytically, the displacement caused by sudden accelerations caused by track anomalies can be represented as two integrations of the acceleration over the time-interval in which the jolts occur. If this time-interval spans from time  $t$  to  $t + \Delta t$  and  $\vec{V} = V_x\hat{x} + V_y\hat{y} + V_z\hat{z}$  is the *velocity* of the train, we will have:

$$\int_t^{t+\Delta t} a_{\hat{x}} dt = V_{\hat{x}}^{\text{jolt}} \longrightarrow \Delta x = \Delta t \cdot V_{\hat{x}}^{\text{jolt}} \quad (6)$$

$$\int_t^{t+\Delta t} a_{\hat{y}} dt = V_{\hat{y}}^{\text{jolt}} \longrightarrow \Delta y = \Delta t \cdot V_{\hat{y}}^{\text{jolt}} \quad (7)$$

$$\int_t^{t+\Delta t} a_{\hat{z}} dt = V_{\hat{z}}^{\text{jolt}} \longrightarrow \Delta z = \Delta t \cdot V_{\hat{z}}^{\text{jolt}} \quad (8)$$

where  $\Delta x/\Delta y/\Delta z$  are the displacements due to jolts in each respective direction. The study conducted by Powell & Palacín, as well as another conducted by Martin & Litwhiler in 2008 [2], suggest that the acceleration & jerk limits above which jolts experienced by passengers should be considered potentially *dangerous* (in addition to being uncomfortable) are  $a = 0.11 - 0.15 [g]$  and  $\mathcal{J} = 0.30 [g/s]$ , respectively.

## 6.4 Averaging Algorithm Implemented in Jolt Analysis

In order to see large acceleration events, we had to look at our data in a different way than it was originally sampled. The LSM9DS1 sampled acceleration data for us at around . Most jolts and jerks experienced by passengers on a moving train are longer than that. Because of this, we decided to average our accelerometer data over 250 millisecond, 500 millisecond, and one second intervals to see if we could get a better look at these events. We did this with Python. With a processed and repaired script, this third program took rapidly sampled accelerometer data, averaged it over a specified time interval, and wrote out a .csv file with the data and its corresponding `latitude`, `longitude` and `millis` data points. It also writes out a jerk value, using a calculation we made to quantify the jolts felt by passengers.

This script uses libraries Numpy, `os.path`, and Pandas. It starts out by reading in the file that were going to be averaging. The only datasets that this program can import are those that whose GPS data have been repaired and that have been processed for mapping. The program then finds the number of lines in the file, and uses Pandas to turn the `millis` column into a series object and then into an array. Next, we establish the indices in the input file of all the data parameters we are interested in using in this stage. For this programs purpose, those are the three dimensional  $\{\hat{x}, \hat{y}, \hat{z}\}$  acceleration columns, the `millis` column, and the `latitude` and `longitude` columns. We also specify the bin width of the time interval we want to sample over. We then find the maximum `millis` value and divide it by the time interval to get a rough estimate of how many data points we will end up with. Rounded to the nearest integer, this value is stored as a variable  $n$ . For all of our analysis thus far, we have been using 250 milliseconds as this time bin interval. Next, zero-value arrays are created through Numpy that are all the size of the variable  $n$ . Because our data points are at irregular time intervals, our data will not fill up this entire array.

This is taken care of with a loop counter variable, which is initialized to be zero right before the main for loop. Our loop looks through every row in the input file. It sets 'now' variables for time, the  $\{\hat{x}, \hat{y}, \hat{z}\}$  acceleration dimensions, latitude and longitude by looking at the indices those values appear in the row, which were specified earlier. It then adds those values together, row by row, until the specified time interval has been reached. Then, it divides that sum by the number of data points taken in that block, saves the value to the data points respective array at the index equivalent to the loop counter, which is only incremented when a full time interval step has been averaged over.

| Processing Phase | Operation(s)   | Filename            |
|------------------|--|---------------------|
| I                | Repair GPS data  | GPS_repair.py       |
| II               | Organize data + minor calculations + prepare for mapping | TrainPy3.py         |
| III              | Calculations of time-averaged acceleration & jerk        | GPS_calculations.py |

Table 4: Brief overview of the processing pipeline through which all data collected during both test-runs and Amtrak rides passes. Note: the filename of step II includes an integer corresponding to the latest version of the program.

At this point, the jerk is to be calculated. Jerk is only calculated when an average has been running for at least two data points. Because of this, it is calculated inside an if statement, where the else statement fills the jerk arrays with NaN characters if jerk is not able to be specified at that time. Jerk is calculated by taking the difference in two subsequent acceleration averages and dividing them by the time bin interval. This calculation is done for all three dimensions of acceleration separately. This allows for a quantitative look into the jolts experienced on a train when acceleration changes quickly. At the end of the loop, all of the sums and incrementer variables are returned to zero. When the loop is over and all the rows have been read, the arrays are trimmed at the loop counter variable, so that none of the zeros present when the array was initialized remain and throw off the data later. Two new arrays are also established, Pythagorean averages of both jerk and acceleration, since they were calculated in three dimensions. Using Pandas, these arrays are all put into a DataFrame with corresponding headers, then written out to a .csv file with a filename that is just the input filename with 'Final' added onto the beginning. In addition to the jerk, as mentioned in the previous section, we also apply a basic integration program under the same averaging algorithm to calculate the displacement of the train over these time-intervals. Figures #12 and #13 show these displacements over quarter-second and half-second intervals (in centimeters) for the Amtrak ride taken on March 9, 2019 with data from Board B. Note the branching shoulders seen on either side of the peak centered at zero; these represent the major displacements caused by jolts during the ride<sup>§</sup>.

## 6.5 Champaign-Urbana MTD Bus Test-Runs

In order to test out the breadboarded and PCB versions of the DAQ, the initial test runs were done on the Champaign-Urbana Mass Transit District. The runs were done on various bus routes with changes in factors like use

<sup>§</sup>In the  $\hat{x}$ -direction specifically, these additional peaks will also have contributions for normal accelerations measured when the train makes planned stops along the route.

### Sample Acceleration Profile for a DAQ Suspended By Springs (MTD Bus)

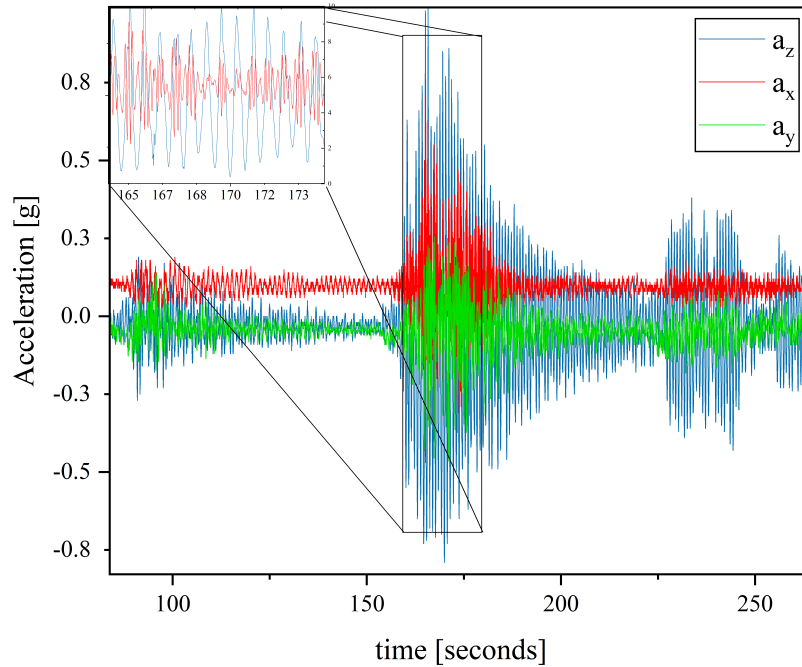
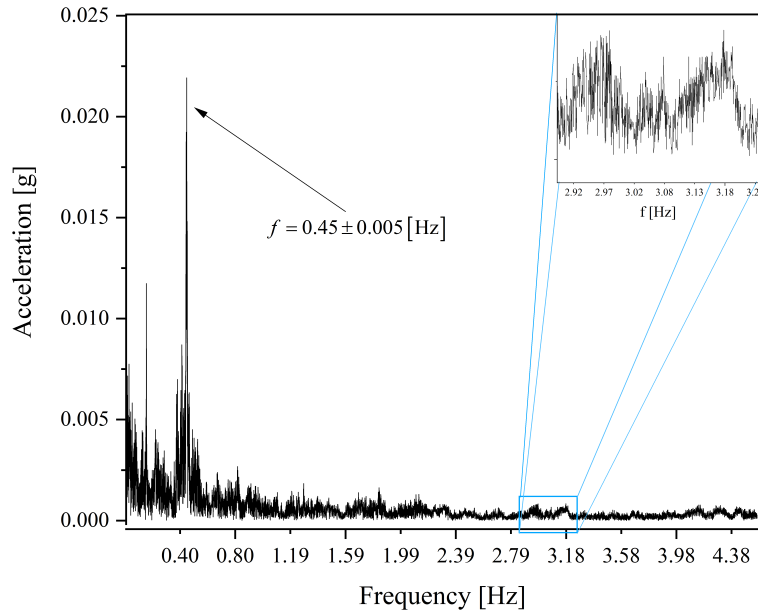


Figure 4: Acceleration profile (all axes) measured by the DAQ suspended by a spring during a test-run on the 5 East Green MTD through Champaign. Note the damped-sinusoidal form characteristic of a spring system. In the enlarged portion located in the Northwest corner, we have omitted the  $\hat{y}$ -axis data in favor of a clearer image.

of springs, using multiple devices in one run and use of GPS with and without antennas. This was done to better understand the setbacks we faced during data collection namely, high-frequency, low-amplitude accelerations, their amplitudes and GPS anomalies (which were eventually discovered to be due to a software error). Overall these runs appeared rougher compared to the Amtrak runs; the overall net-acceleration values averaged higher for each MTD test-run. To deal with unexpected low-amplitude, high-frequency vibrations measured by the accelerometer on the bus, we attempted to take a ride with two PCB devices suspended by springs from the handrail on the bus. Sample 3-axis acceleration data from one of these PCBs is plotted in Figure #4. MTD Acceleration data showed noise with high-frequency low-amplitude accelerations recorded along all axes along with sudden bursts of high acceleration amplitudes. However, events like the bus slowing to a halt or executing a turn have been clearly discernible in the 2-dimensional plots of acceleration & speed, as well as on the maps made in Tableau.

Plotted in Figure #5 is the Fourier-Transform of the data shown in Figure #4. The dominant peak at  $f = 2.80 \pm 0.005$  [Hz] is the natural-frequency of the spring used to suspend the PCB from the handrail of the MTD bus. Evidently the high-frequency, low amplitude oscillations initially measured on MTD test-runs were successful decoupled from the DAQ through the addition of the spring.

Fourier Transform:  $a_{\text{net}}$  from MTD Ride w/ Board Suspended from Spring



Fourier-Transform:  $a_{\text{net}}$  from MTD Ride w/o Spring

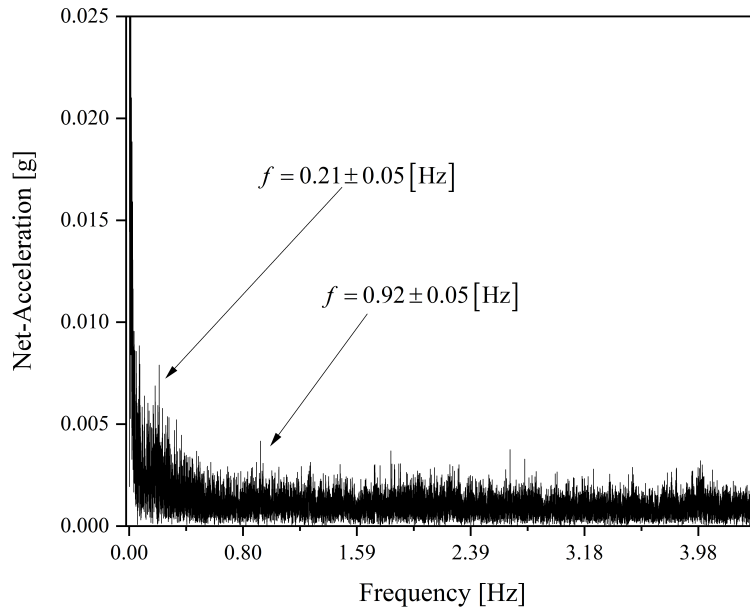


Figure 5: Fourier Transforms of the same time-domain data plotted in Figure #4, as well as a frequency spectrum measured by a DAQ aboard the same bus not without coupling to a spring. Note the overwhelmingly dominant peak at what is likely the natural frequency of the spring in the uppermost FFT.

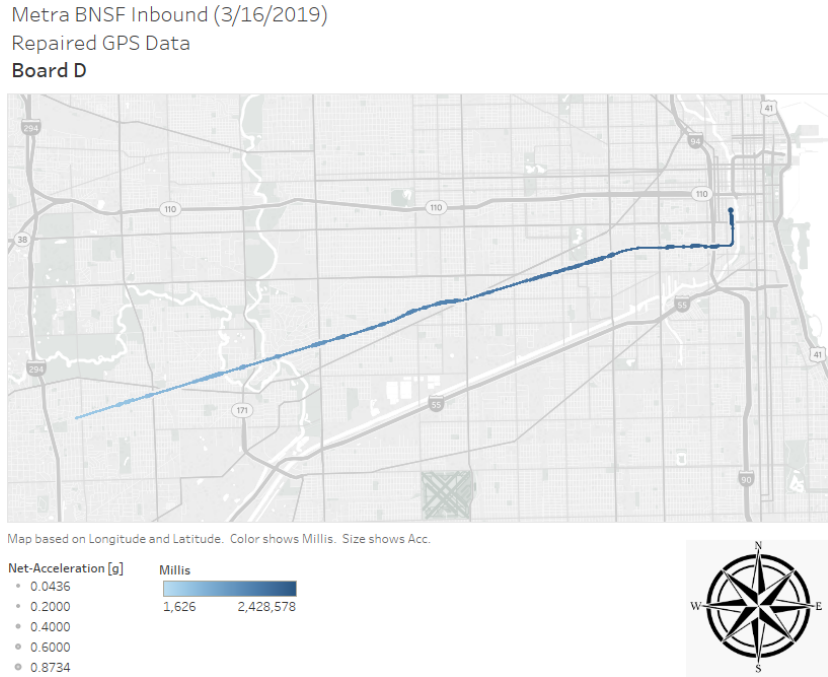


Figure 6: Map of Chicago Metra inbound trip from 3/16/2019. Generated with processed GPS data using Tableau Desktop.

## 6.6 Chicago METRA Train Test-Runs

Additional test-runs were carried out on the Chicago Metra train lines, specifically the Southwest Service Route from Palos Heights, Il. to Chicago Union Station. These tests were conducted in order to compare the acceleration profiles of Amtrak railways to commuter lines. Since the Metra trains make a large number of successive stops, as opposed to Amtrak which runs more or less continuously for long periods of time, we were able to verify that the accelerations measured in the  $\pm\hat{x}$ -directions by the LSMD9S1 matched the *speeds* read-out by the GPS unit. Figure #8 shows the acceleration and speed profiles of one of these test-runs. The frequency spectra were generated via Fourier analysis of the net-acceleration profiles for both the inbound and outbound Metra rides. Significant peaks are visible at  $f \approx 0.24$  [Hz] for both the inbound and outbound rides. It is possible that this low-frequency vibration was generated by some form of electrical equipment aboard the train. In addition, prominent peaks in the FFT spectrum appear in the Metra data at  $f = 1.29 \pm 0.03$  [Hz] &  $f = 2.14 \pm 0.05$  [Hz] for the inbound and outbound rides, respectively. These do not correspond to any known electrical or mechanical systems that we were able to identify during our investigations.

The GPS data for the inbound Metra run are mapped in Figure #6. The GPS data is accurate for these rides; it follows the *Southwest Service* Metra line all the way to Chicago from the Southwest suburbs, losing the satellite fix while entering the large concrete tunnel at Chicago Union Station. These rides served as useful test-runs as they provided us an opportunity to obtain data to compare with the measurements made on Amtrak railways. They also allowed us to test how well the measurements of acceleration - made by the LSM9DS1 - aligned with the measurements of speed made separately by the GPS unit (see: Figure #8).



Fourier Transform:  $a_{\text{net}}$  for Metra Chicago (Inbound)

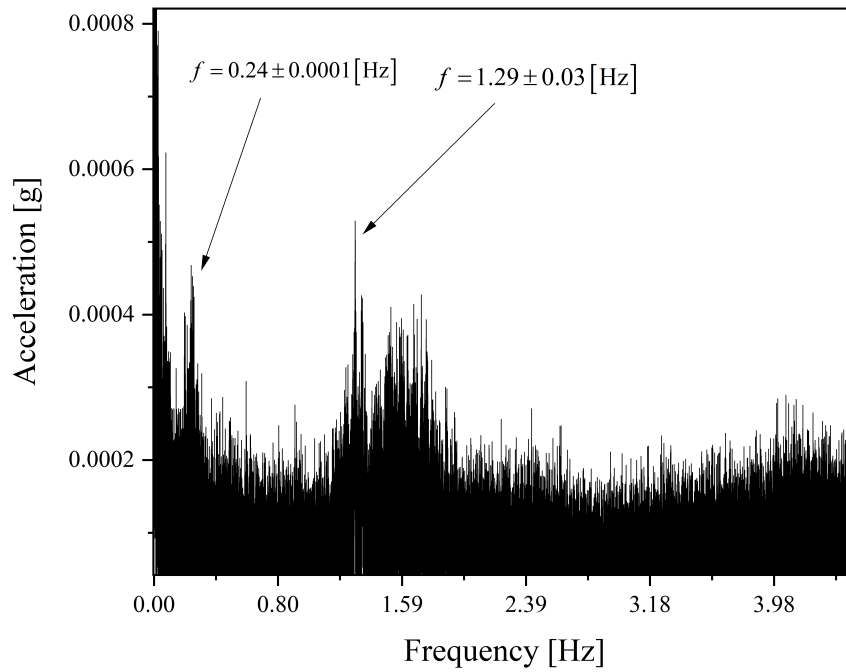


Figure 7: Fourier Transform of Metra inbound data.

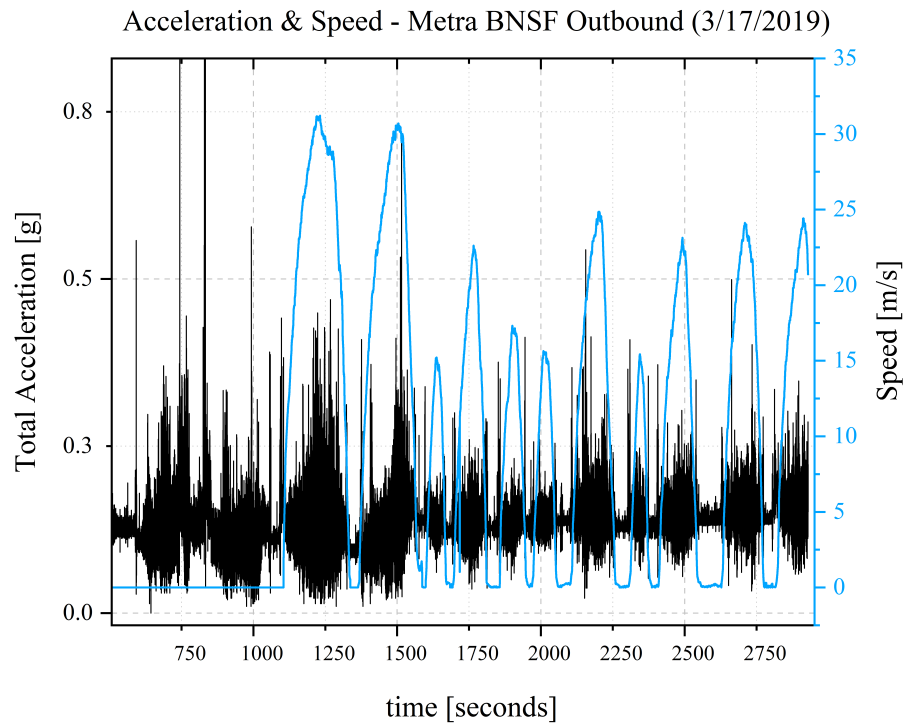


Figure 8: Profile of the net-acceleration experienced aboard an inbound Metra train from Union Station to the Southwest Chicago suburbs. The speed of the train is overlaid in blue.

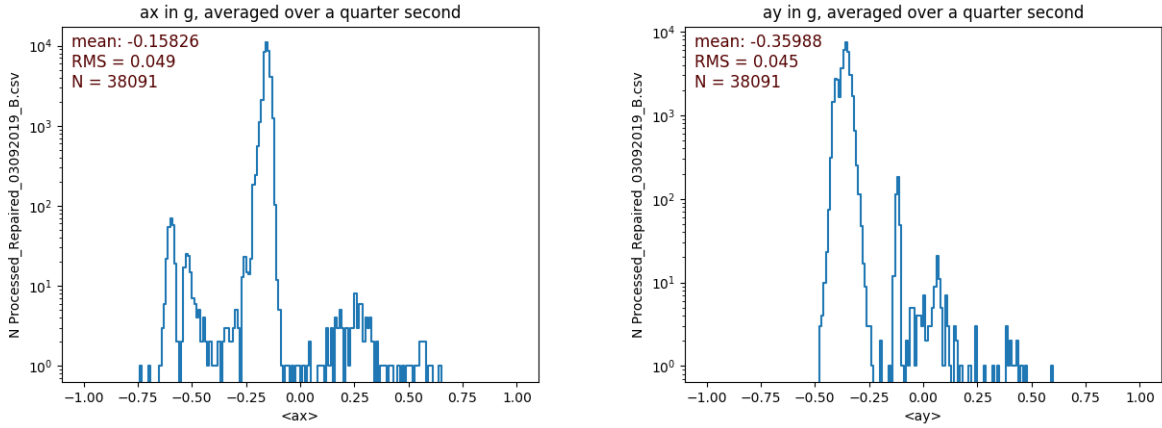


Figure 9: Histograms time-averaged acceleration data in the  $\hat{x}$  &  $\hat{y}$  directions for the Amtrak Champaign-Chicago runs.

## 6.7 Amtrak Runs: Champaign, IL. to Chicago IL.

The Amtrak Illinois Service line was taken out of the Illinois Terminal in Champaign, IL. to Chicago Union Station on March 9, 2019 and then back to Champaign on March 10, 2019. For this run, *four devices were present aboard the train*; this was arranged so that as complete a dataset as possible could be obtained. This also allowed us to examine the behavior of each of the boards with respect to each other, in order to isolate any unexpected or anomalous behavior by each board. The data for the net-accelerations measured by *each board*<sup>‡</sup> are plotted in Figure #16. Upon inspection of Figure #16, it can be seen that the acceleration profiles measured by *each board* match each other to within a few hundred milliseconds, consistently registering the same jolts of acceleration. There are a few notable exceptions however; examining the profile of Board C (*orange*) in figure #16 for instance, we see a jolt registering near  $a_{\text{net}} \approx 0.2$  [g] at  $t \approx 9.1 \times 10^6$  [ms] = 151.7 [minutes], whereas none of the other boards appear to have measured this jolt at all. Events like these are usually marked by the group member conducting the data collection however this particular jolt did not appear in the record. It is possible that the individual board was jostled by movement of passengers in the train as it was closest to the aisle.

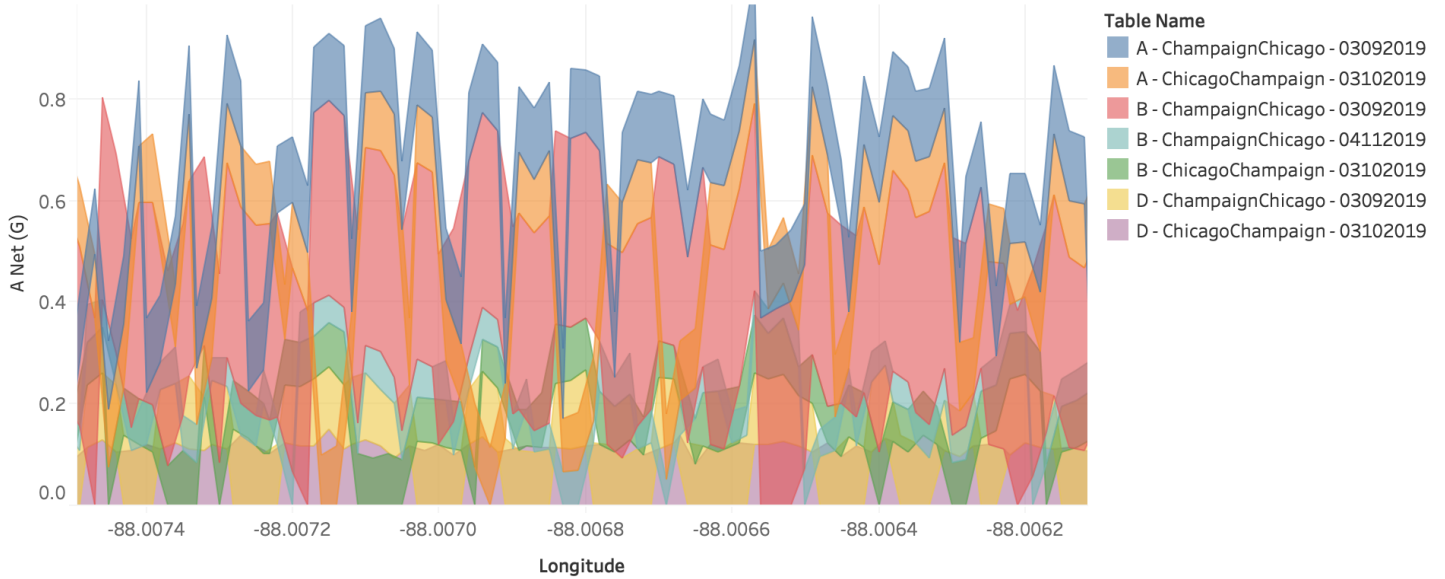
<sup>‡</sup>The four PCBs were designated *A, B, C, & D* in order to keep track of which PCB belonged to which group member. In order, these corresponded to K. Vaidya, L. Gregory, K. Na, and D. MacLean. B. Franklin's board was unavailable for this run.

Most rides on Amtrak taken during our semester-long investigations were carried out on this Illinois Service route. In addition to the four-board rides conducted on March 9 & 10, additional rides were taken on March 13 and April 11. Unfortunately, these rides did not have more than three DAQ devices present for data collection, however the repeated trips over the same stretch of track allow us to better understand which major jolts recorded in the data are due to legitimate track anomalies. A number of major jolts exceeding  $a = 0.3 [g]$  (in one or more direction measured by the accelerometer) were registered at effectively the same latitude/longitude coordinates across the series of rides. A net-acceleration profile accumulated from the measurements made by all boards over all rides through this route is shown in Figure #15, in addition to a jolt-density map showing the locations of rougher segments of track in which larger net-accelerations were measured. The root mean-square acceleration measured on this series of rides, averaged across all runs and boards present for each run was calculated to be  $a_{\text{rms}} = 0.15 \pm 0.03 [g]$ . Averaging the  $\hat{x}$  and  $\hat{y}$  acceleration data collected on the March 9, 2019 ride over a fixed quarter-second (250 milliseconds) interval, we produce the histograms seen in Figure #9. As expected, we see a peak centered near  $a_{\hat{x}} = a_{\hat{y}} = 0$ , however we also see additional “shoulder” peaks near  $a_{\hat{x}} = \{-0.55 \pm 0.05, 0.25 \pm 0.01\} [cm]$  and  $a_{\hat{y}} = 0.35 \pm 0.05 [g]$  (where the  $a_{\hat{y}}$  value has been shifted to such that the dominant peak lies at zero; this shift was likely due to the fact that the board was lying at an angle on the seat of the train). These shoulder-peaks indicate additional accelerations experienced aboard the train the form of the lurches/jolts we are concerned with. It is important to note, however, that the  $a_{\hat{x}}$  acceleration data will also reflect a non-negligible contribution from any planned stops made by the train at intermediate train stations along the way. For the case of the Illinois Service routes, these stops are located at Rantoul, Gilman, Kankakee, and Homewood Illinois. The specific latitude/longitude coordinates of each stop are tabulated for reference in Table #5. Turning our attention to the displacement (magnitudes) caused by the lurching of the train, we examine the set of histograms<sup>¶</sup> shown in Figures #12, #13, & #14. These data show that the displacements in the  $\hat{x}$  &  $\hat{y}$  directions caused by major jolts tend to be on the order of  $\Delta x = 20 \pm 10 [cm]$  and  $\Delta y = 15 \pm 10 [cm]$  while averaging over a quarter-second time interval and  $\Delta x = 50 \pm 15 [cm]$  and  $\Delta y = 55 \pm 15 [cm]$  while averaging over a half-second time interval. Recall that normal accelerations due to stops will tend to skew the  $\hat{x}$  acceleration to higher values. Finally, applying the averaging algorithm over a one-second time interval, we see that the average displacements caused by major jolts are  $\Delta x = 250 \pm 55 [cm]$  and  $\Delta y = 150 \pm 50 [cm]$ .

---

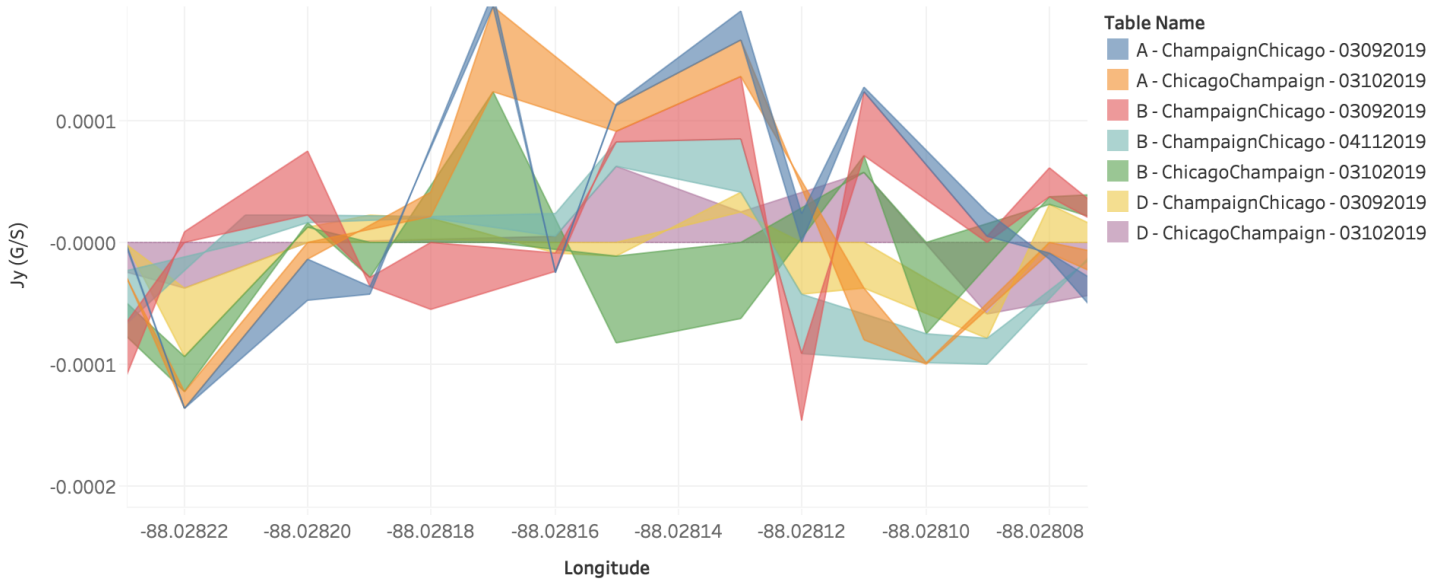
<sup>¶</sup>These histograms are generated using the `GPS_calculations.py` script written by Professor George Gollin.

### Acceleration Over Longitude (Zoomed)



Longitude vs. A Net (G). Color shows details about Table Name. The view is filtered on Table Name, which excludes D - ChicagoMilwaukee - 03202019.

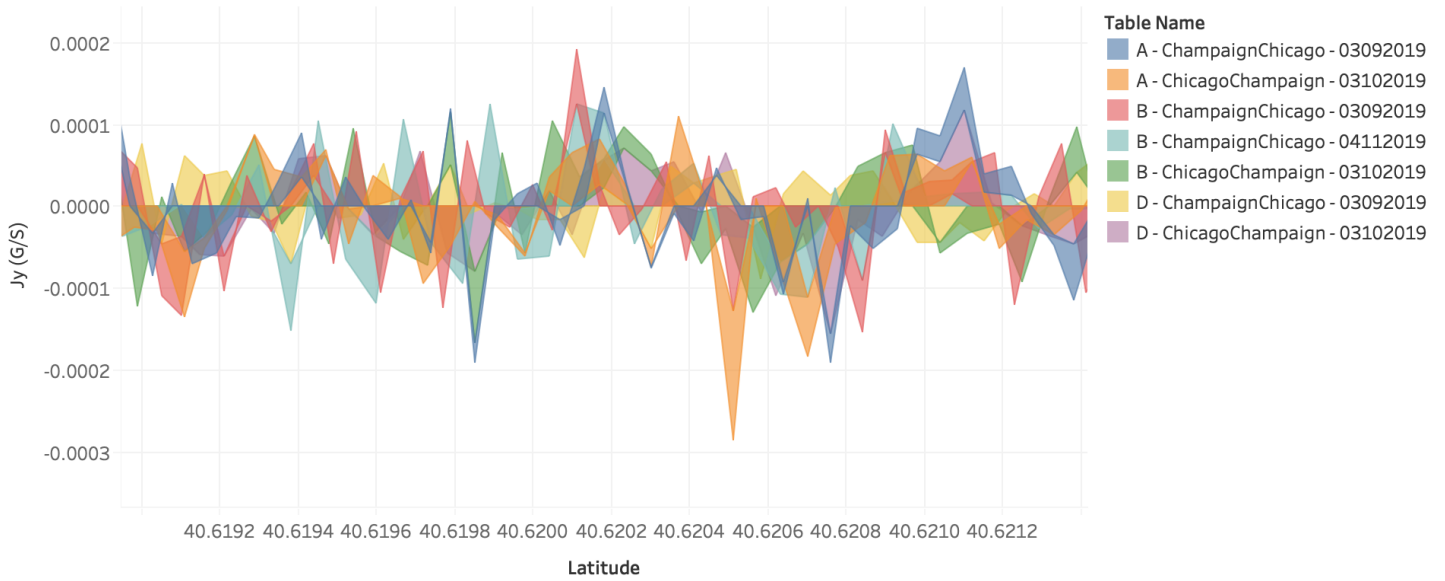
### Jerk Over Longitude (Zoomed)



Longitude vs. Jy (G/S). Color shows details about Table Name. The view is filtered on Table Name, which excludes D - ChicagoMilwaukee - 03202019.

Figure 10: Plots of the acceleration and jerk on the Chicago-Champaign Amtrak route.

### Jerk Over Latitude (Zoomed)



Latitude vs. Jy (G/S). Color shows details about Table Name. The view is filtered on Table Name, which excludes D - ChicagoMilwaukee - 03202019.

### Density Map of Jerk (Zoomed)



Map based on Longitude and Latitude. Color shows J Net (G/S). Details are shown for Table Name.

Figure 11: Magnified plot (vs. latitude) and jolt-density map of jerk measured on Chicago-Champaign Amtrak route. This magnified region was selected as it corresponds to a rough segment of track between Gilman and Kankakee, Illinois.

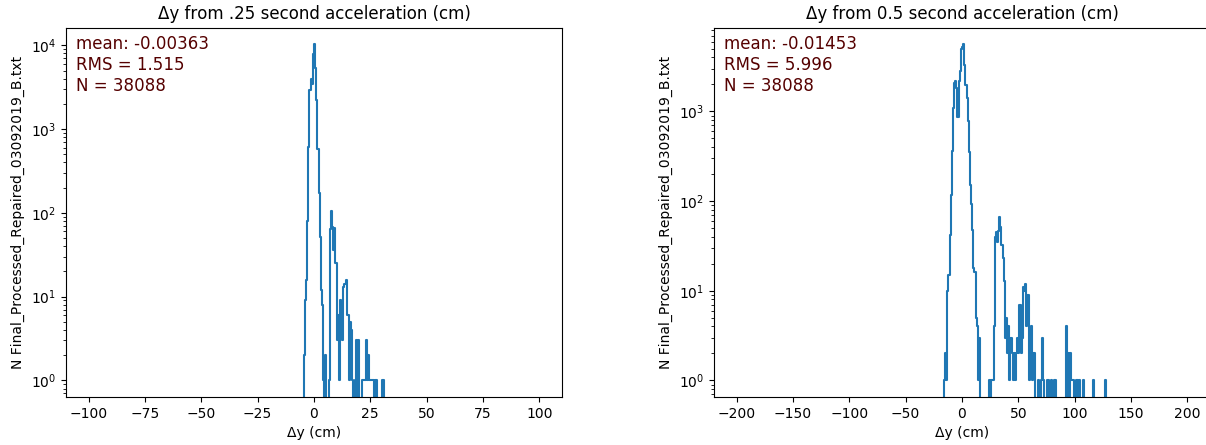


Figure 12: Displacements in the  $\hat{y}$ -direction from quarter-second and half-second accelerations measured on Amtrak Illinois Service Route between Chicago and Champaign.

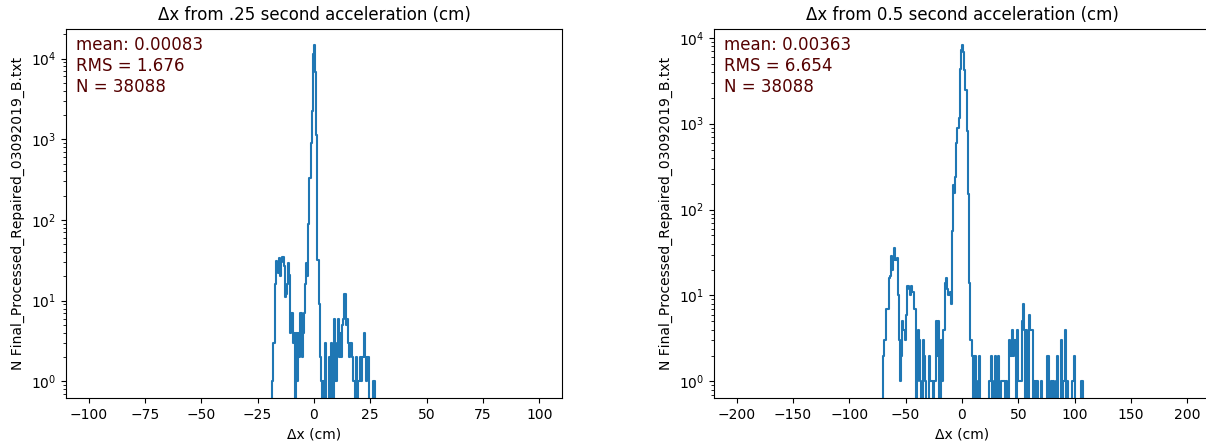


Figure 13: Displacements in the  $\hat{x}$ -direction from quarter-second and half-second accelerations measured on Amtrak Illinois Service Route between Chicago and Champaign.

Examining the mapped data for the Champaign-Chicago Amtrak rides, it can be seen that there are indeed major jolts/rough segments of track even when the planned stops are accounted for. The roughest segments of track in which the highest numbers of jolts were registered from ride-to-ride were undoubtedly between the Kankakee and Homewood, IL. stops and through the entire Chicagoland area into Union Station after the Homewood stop. The track passing through open stretches of central Illinois was generally smoother, with the most jolts occurring in an approximately ten mile-long stretch leading into the Kankakee stop. All of the positions corresponding to scheduled stops display large spikes on both the Acceleration vs. Longitude and Jerk vs. Longitude graphs. However it seems that there is an overall steady amount of noise most likely due to the vibration of the train itself. In addition to this there are some areas, offset from the positions in which the train comes to a stop, such as around longitudinal positions -88.20, -88.125, and -87.975 for example, that also show some signs of what could potentially be an area where the train experiences a large amount of jerk. These spikes tend to range from 0.5 to -0.5 [g/s]. Let it be known

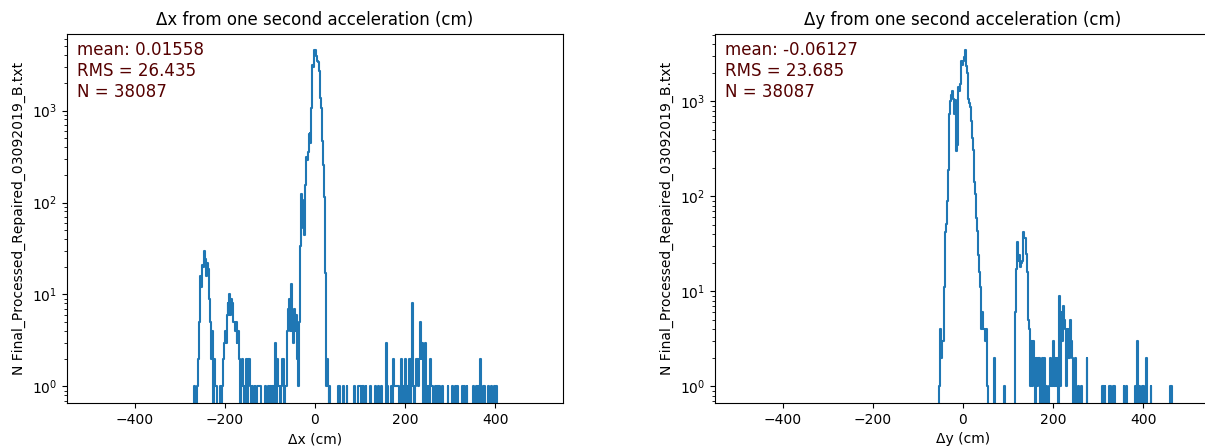


Figure 14: Displacements in the  $\hat{x}$  &  $\hat{y}$ -directions from one second accelerations measured on Amtrak Illinois Service Route between Chicago and Champaign.

that these similar areas of jerk appear frequently throughout the ride and are visible on the plots.

| Stop           | Latitude Coordinate | Longitude Coordinate |
|----------------|---------------------|----------------------|
| Champaign, IL. | 40°06'56.5" N       | -88°14'28.4" W       |
| Rantoul, IL.   | 40°18'40.1" N       | -88°09'32.5" W       |
| Gilman, IL.    | 40°45'09.6" N       | -87°59'52.7" W       |
| Kankakee, IL.  | 41°07'08.9" N       | -87°51'56.7" W       |
| Homewood, IL.  | 41°33'44.7" N       | -87°40'07.2" W       |
| Chicago, IL.   | 41°52'43.0" N       | -87°38'21.0" W       |

Table 5: Stops along the Illinois Service Route from Champaign to Chicago, Illinois. The GPS coordinates reflect the specific locations of the Amtrak train stations, with the terminus in Chicago corresponding to Union Station.

## 6.8 Amtrak Run: Chicago, IL. to Milwaukee, WI.

The Hiawatha 333 & 336 trains from Chicago Union Station to Milwaukee’s Intermodal Station were taken on March 20, 2019. The train made stops in Glenview, IL., Sturtevant, WI. and The Milwaukee Airport. Profiles of the *net* acceleration and train-speed are shown in Figures #19 & #17. For this run, only a single board (Board D) was available for data collection. As such, there is only one dataset available rather than four for comparative analysis. Examining the acceleration profile for this ride, we find that the ride was undoubtedly rougher near the Chicagoland area, with numerous jolts registering at or above  $a_{\text{net}} = 1.5 [g]$ . Fourier Transforms of the Amtrak data for rides between Milwaukee and Chicago reveal a prominent spike at  $f = 2.02 [\text{Hz}]$  for *both* trips (see: Figure #19). Unfortunately, due to time and logistical constraints, only one ride was taken over this route. As such, the major jolts registered by the DAQ are harder to verify as legitimate accelerations caused by track anomalies. Calculation of the root mean-square acceleration over this rides yields the value  $a_{\text{rms}} = 0.17 \pm 0.02 [g]$  which we interpret to correspond to a slightly rougher ride than those taken between Chicago and Champaign, which had an RMS acceleration of  $a_{\text{rms}} = 0.15 \pm 0.03 [g]$  on average.

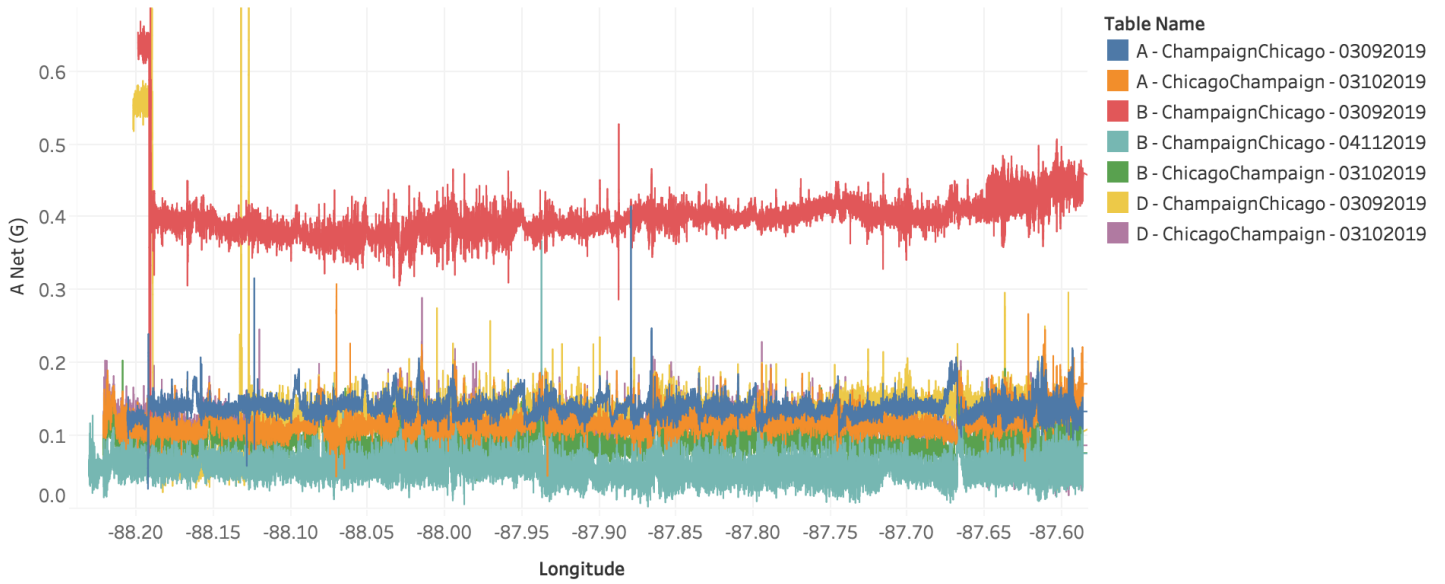
As before, we must take into account the planned stops made along the Hiawatha Service route, as the accelerations into and out of these stops will undoubtedly contribute to the measurements made by the DAQ. These stops occur at Glenview, Illinois then Sturtevant, the General Mitchell International Airport just outside of Milwaukee, and Intermodal Station in downtown Milwaukee all in Wisconsin. The roughest segments of track between Chicago and Milwaukee were found to lie closer to the Chicagoland area. Past the state border into Wisconsin, the track smoothed-out substantially.

| Stop                                | Latitude Coordinate | Longitude Coordinate |
|-------------------------------------|---------------------|----------------------|
| Chicago, IL.                        | 41°52'43.0" N       | −87°38'21.0" W       |
| Glenview, IL.                       | 42°04'30.4" N       | −87°48'20.5" W       |
| Sturtevant, WI.                     | 42°43'05.68" N      | −87°54'21.6" W       |
| General Mitchell Int'l Airport, WI. | 42°56'26.2" N       | −87°55'28.4" W       |
| Milwaukee, WI.                      | 43°02'04.1" N       | −87°55'04.9" W       |

Table 6: Stops along the Hiawatha Service Route between Chicago and Milwaukee, Wisconsin. The GPS coordinates reflect the specific location of the Amtrak train stations, with the terminus in Chicago corresponding to Union Station.



### Acceleration Over Longitude (Full Data)



Longitude vs. A Net (G). Color shows details about Table Name. The view is filtered on Table Name, which excludes D - ChicagoMilwaukee - 03202019.

### Net Acceleration (All Data, Full Track)



Figure 15

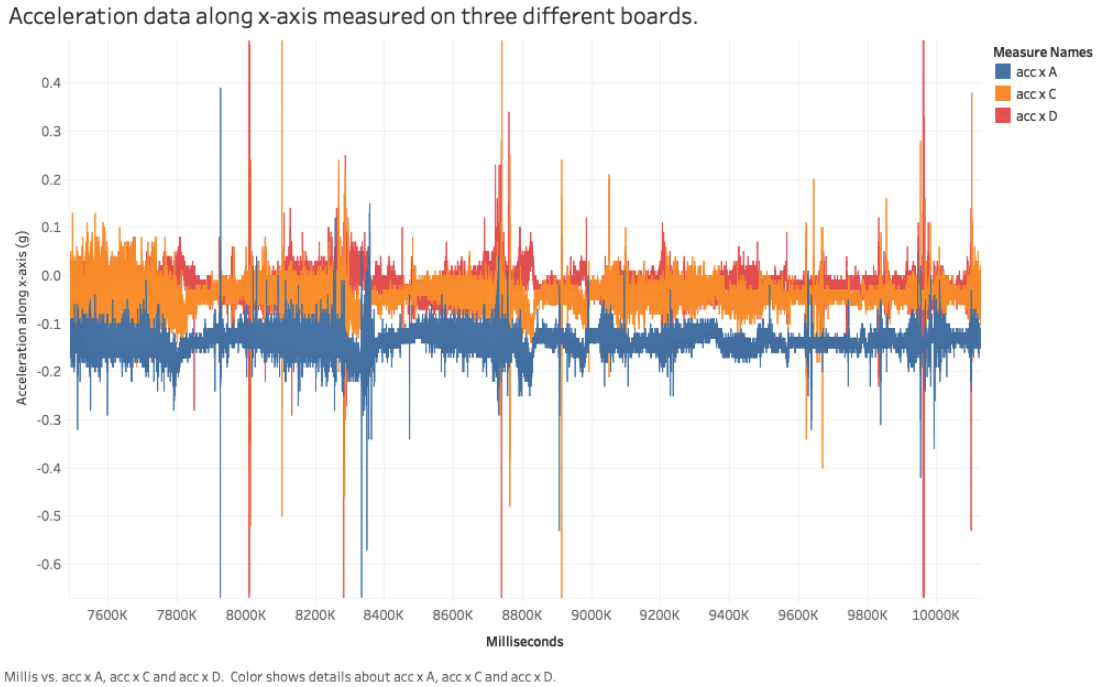


Figure 16: Net-acceleration profiles measured by three of the four DAQ board present on the 3/9/2019 & 3/10/2019 Amtrak runs between Champaign and Chicago.

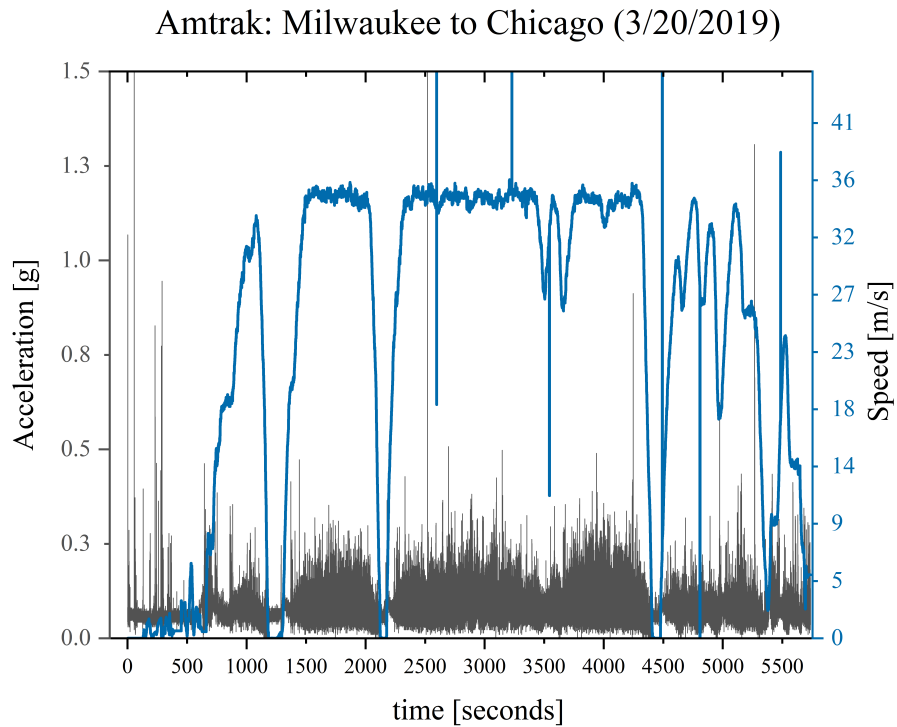


Figure 17: Profile of the net-acceleration experienced aboard an Amtrak train from Milwaukee, WI to Chicago, IL, with the speed of the train overlaid.

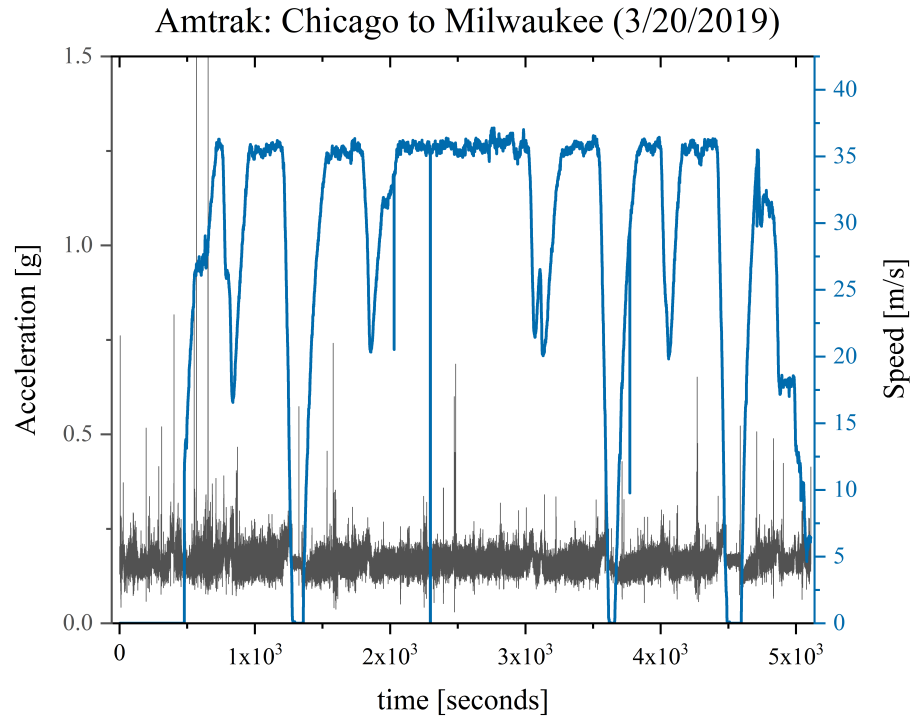


Figure 18: Profile of the net-acceleration experienced aboard an Amtrak train from Chicago, Il. to Milwaukee, Wi. with the speed of the train overlaid.

Fourier Transform:  $a_{\text{net}}$  for Amtrak Run (Chicago to Milwaukee)

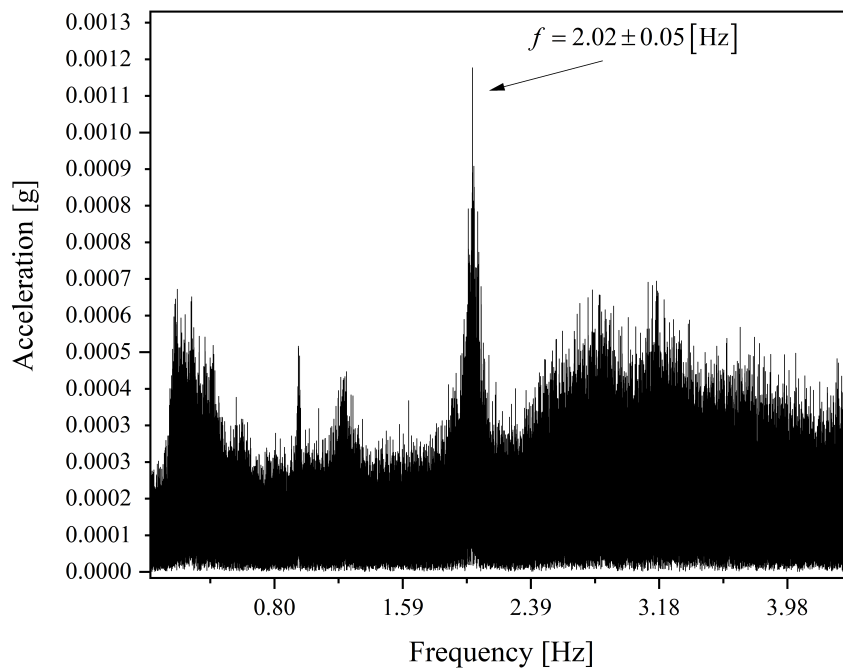
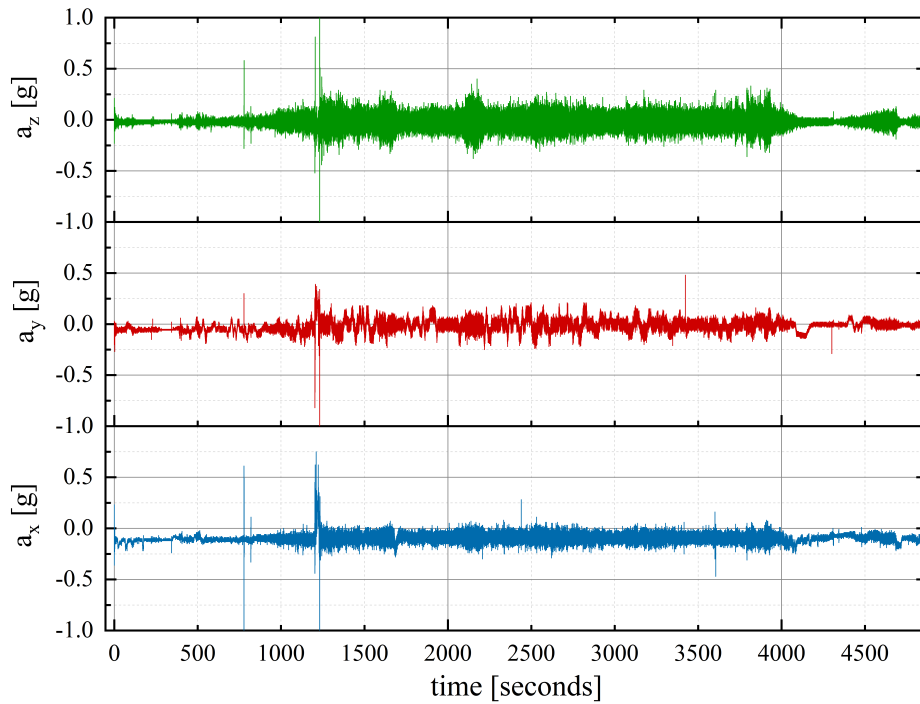


Figure 19: Fast Fourier-Transform (FFT) of acceleration profile measured on the Amtrak route from Chicago to Milwaukee. Note the prominent spike at  $f = 2.02 \pm 0.05$  [Hz].

## 6.9 Rides on Italian Inter-City Trains

During the course of our semester-long investigations into rail anomalies, we were presented with a unique and serendipitous opportunity to gather additional comparative data. It so happened that one of our group members, Lauren Gregory, was scheduled to take a trip to Italy during the Spring Break period in March 2019. At this point in the semester, the PCB versions of the DAQ were already assembled and running smoothly and fortunately she was able to bring her DAQ along with her. Several datasets were gathered on Italian inter-city trains and compared to the data gathered on American Amtrak routes in order to ascertain the differences between the two nations' rail infrastructure. We began this project under the assumption that Italys intercity train lines would be a smoother ride than their American counterparts. While this was largely true from a subjective riders point of view on the train, the data told a different story. As seen in Figure #20, riders on the Italian train actually experienced root mean-square (RMS) net-acceleration than those aboard Amtrak, with good deal more noise present in the data as well. The average acceleration for the Italian train was  $a_{\text{rms}} = \sqrt{\langle a_{\text{net}}^2 \rangle} = 0.12 \pm 0.005$  [g] whereas the average acceleration for a typical Amtrak train is  $a_{\text{rms}} = 0.16 \pm 0.03$  [g], however the largest jolt measured aboard an Italian train was  $a_{\text{net}}^{\text{max}} = 3.052 \pm 0.001$  [g] while the most intense jolt felt on the Chicago - Milwaukee trip was  $a_{\text{net}}^{\text{max}} = 1.933 \pm 0.002 \pm 0.001$  [g]. There are a few possible explanations for this. First, the trains were traveling at drastically different speeds. The Italian train traveled at a cruising max speed of about 140 miles per hour (62.59 meters per second), while the Amtrak traveled at a max cruising speed of 67 miles per hour (29.84 meters per second). We hypothesize that any problems on the track could be amplified by the faster motion and shown in the data as higher acceleration. The higher velocity of the Italian trains may also account for the higher noise-level that was measured. During MTD test-runs, undoubtedly the lowest-speed contexts in which data collection was carried out, the noise-level was lowest. Plots of the three-axis acceleration profile as well as a Fourier Transform of the *net*-acceleration from this ride are shown in Figure #20. The jolt-density maps produced for the Italian train rides are shown in Figure #21.

### Italian Railway from Rome to Florence: Three-Axis Acceleration Profile



### Fourier Transform: Italian Inter-City Train from Rome to Florence

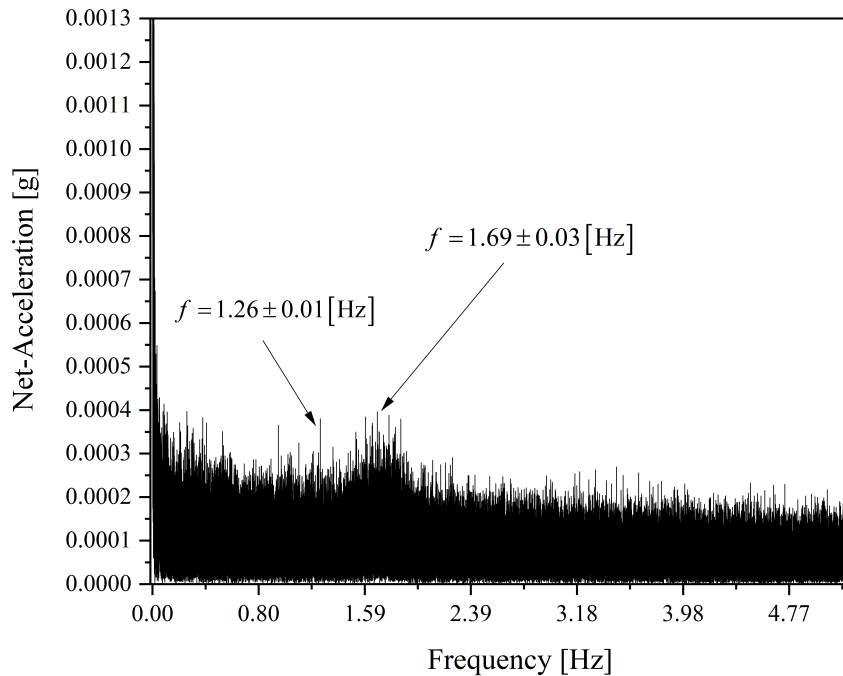
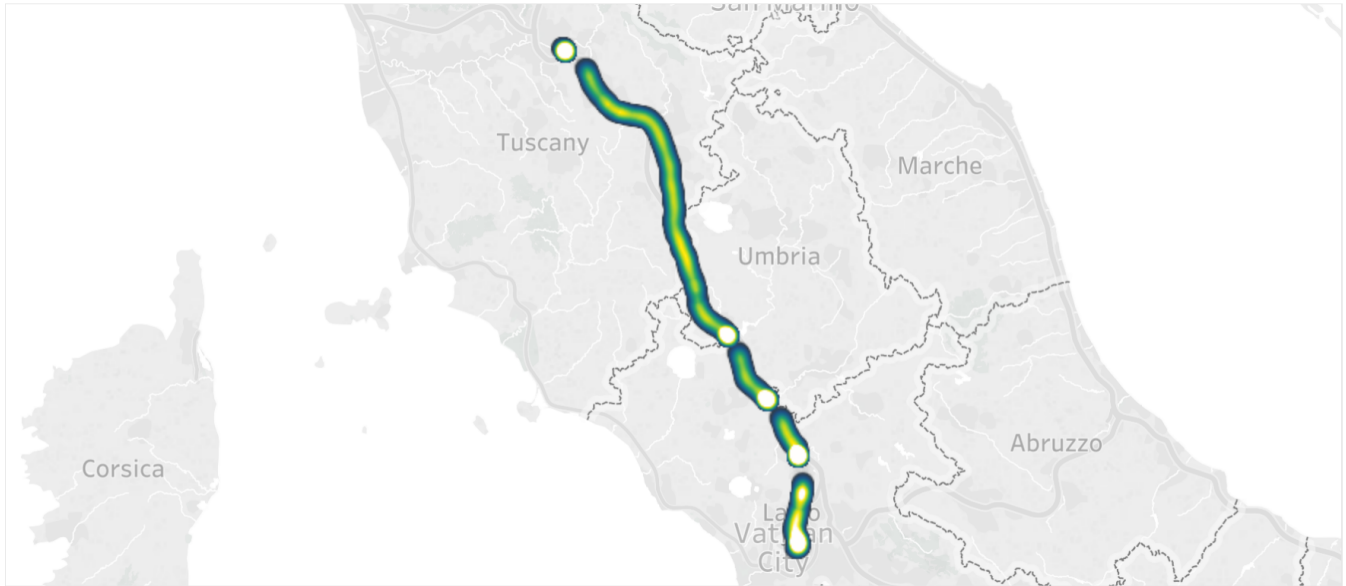


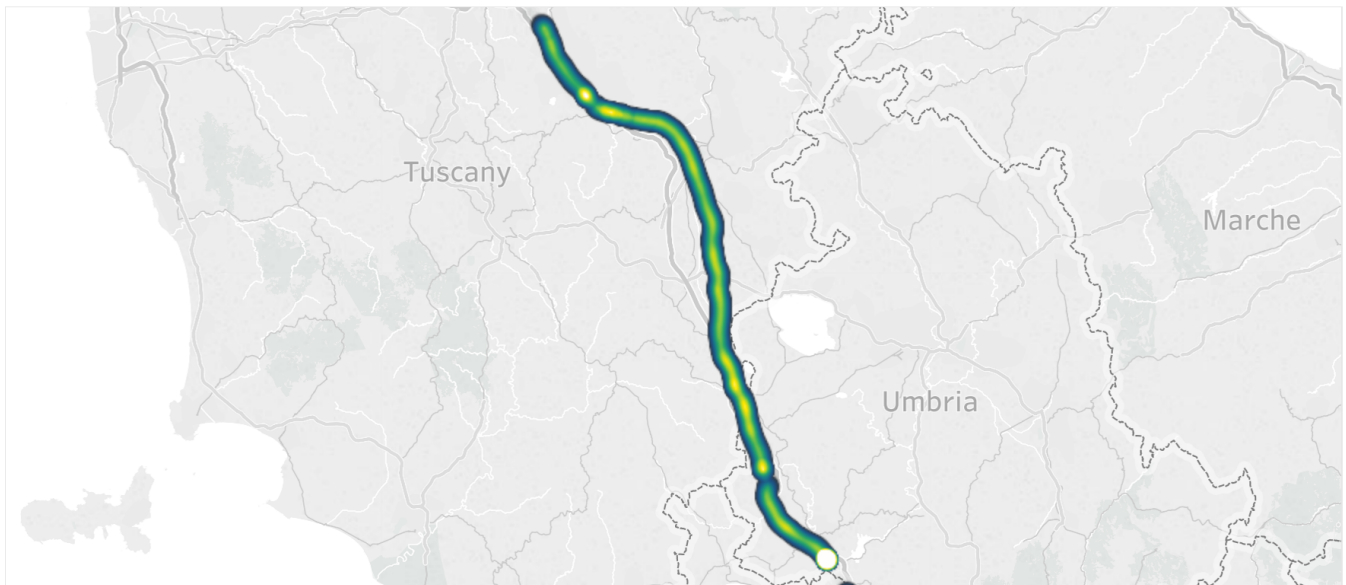
Figure 20: Acceleration profile and Fourier Transform (net-acceleration) of ride on Italian inter-city train from Rome to Florence.

## Florence to Rome



Map based on Longitude and Latitude. Color shows Jerk.

## Florence to Rome (Zoomed)



Map based on Longitude and Latitude. Color shows Jerk.

Figure 21: Jolt-density maps produced with the Italian railway datasets.

## 7 Conclusions

Throughout the course of our analysis of Amtrak railways, we have found that the tracks between both Chicago and Champaign Illinois as well as Chicago and Milwaukee, Wisconsin caused numerous jolts registered by the accelerometer as exceeding 2 [g] of rapid-burst acceleration. When compared to the data collected on Italian inter-city trains, we find that the Amtrak railways cause far more high-amplitude jolts with larger displacements than do the European railways. Fourier Analysis of the data collected on Amtrak revealed a prominent peak at frequency  $f = 2.02$  [Hz] for both trips between Chicago and Milwaukee. We ultimately were unable to match this frequency with any known mechanical or electrical apparatus aboard the train that could cause such a signature, however we do not definitively discount this possibility. It was found that the room-mean-square accelerations felt on Italian inter-city trains was *larger* than those experienced on Amtrak railways. We attribute this to the fact that the Italian trains moved at a significantly higher rate of speed than the American Amtrak trains (almost twice as fast, at certain points during the ride). Analysis of the jerk calculated based on the acceleration data for all Amtrak rides has led us to conclude that the region between Kankakee and Homewood, Illinois is the roughest stretch of track along the Illinois Service Route (see: jolt-density map in Figure #15). We find that the displacements caused by the major jolts in the track are heavily dependent on the time-interval over which these jolts occur. A quarter-second jolt could be enough to displace the train car up to 15 centimeters in the  $\hat{y}$  (side-to-side) direction, whereas a half-second jolt in the  $\hat{y}$ -direction may cause the train to lurch up to 50 centimeters.

We have chosen not to include any tables of raw datasets in this work as they are generally massive (Amtrak datasets typically approach a million data-points), however we are glad to provide<sup>†</sup> any party interested with copies of any raw or processed datasets we have heretofore gathered. This project is highly amenable to further work and should by no accounts be considered exhaustively complete. Our rides on Amtrak railways covered only a fraction of the national network. In addition, only the route between Champaign - Chicago was traveled multiple times and with more than one board present aboard the train. The Chicago - Milwaukee route should be re-traversed with multiple DAQ devices aboard. We also recommend that future studies of similar nature prioritize the *Northeastern Corridor* Amtrak line, as it is the busiest of all Amtrak routes as of 2019. We hope that our efforts throughout the course of this investigation will not only serve to highlight any issues currently existing in American infrastructure, but also to inspire future solutions based on the methods we have used to analyze the state of Amtrak railways.

## References

- [1] J. P. Powell, R. Palacín. *Passenger Stability Within Moving Railway Vehicles: Limits on Maximum Longitudinal Acceleration*. Urban Rail Transit (2015) 1(2):95103. DOI: 10.1007/s40864-015-0012-y
- [2] D. Martin, D. Litwhiler. *An Investigation of Acceleration and Jerk Profiles of Public Transportation Vehicles*. Pennsylvania State University-Berks. American Society for Engineering Education, 2008.

---

<sup>†</sup>The email addresses of the authors in the order they are presented on the first page: lmg2@illinois.edu, dmaclea2@illinois.edu, kvaidya2@illinois.edu, bifrank2@illinois.edu, kna3@illinois.edu.

- [3] McBride, James. *The State of U.S. Infrastructure*. Council on Foreign Relations, 12 Jan. 2018, [www.cfr.org/backgrounder/state-us-infrastructure](http://www.cfr.org/backgrounder/state-us-infrastructure). Accessed 22 Mar. 2019.
- [4] Gollin, Prof. George D. *Course Packet*. Physics 398 DLP. Spring 2019.
- [5] Amtrak Publication. *Fiscal Year 2018 Company Profile*. Accessed March 18, 2019.
- [6] American Society of Civil Engineers. *2017 Infrastructure Report Card: Rail*. p71 - p75. Accessed March 25, 2019.
- [7] Micro SD Card Breakout Board Tutorial. Lady Ada. Adafruit Industries.
- [8] INA219 Current Sensor Breakout datasheet. Lady Ada. Adafruit Industries.
- [9] Maxim Integrated. *Extremely Accurate I2C-Integrated RTC/TCXO/Crystal*. Datasheet.
- [10] NXP Semiconductors N.V. I2C-bus specification and user manual.
- [11] LSM9DS1 Accelerometer + Gyro + Magnetometer 9-DOF Breakout datasheet. Lady Ada. Adafruit Industries.
- [12] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V datasheet. Atmel.
- [13] Ultimate GPS breakout board datasheet. Lady Ada. Adafruit Industries.
- [14] FGPMMPA6H GPS Standalone Module. Datasheet. GlobalTop Technology Inc. Revision: V0A



## 8 Appendix I: Dilution of Acceleration Data by Earth's Gravity

As mentioned in section #6, it is necessary to remove the constant 1 [g] acceleration due to Earth's gravity present in all  $\hat{z}$ -oriented acceleration profiles. If the influence of gravity is left in the  $a_{\hat{z}}$  data for a given train or bus ride, it will effectively *dilute* any anomalous accelerations present in the net-acceleration profile. We calculate the net-acceleration profiles using the three Cartesian axes of acceleration data measured by the DAQ by adding them together in quadrature; reiterating Equation #1:

$$a_{\text{net}} = \sqrt{a_{\hat{x}}^2 + a_{\hat{y}}^2 + a_{\hat{z}}^2} \quad (9)$$

However, if we re-write the vertical acceleration  $a_{\hat{z}}^{\text{tot}}$  as the sum of the acceleration due to gravity *and* due to any additional accelerations  $a_{\hat{z}}$  (i.e. jolts or roughness in the ride itself), we will have:

$$a_{\hat{z}}^{\text{tot}} = (a_{\hat{z}} + g) = g \left( 1 + \frac{a_{\hat{z}}}{g} \right) \quad (10)$$

So, returning to the *net* acceleration, we obtain:

$$a_{\text{net}} = \sqrt{a_{\hat{x}}^2 + a_{\hat{y}}^2 + g^2 \left( 1 + \frac{a_{\hat{z}}}{g} \right)^2} \quad (11)$$

For a jolt due, for instance, to a rough section of track on an Amtrak railway, we see accelerations registered typically in the range  $a_{\text{net}} \approx 0.5$  [g] to  $a_{\text{net}} \approx 3$  [g]. Let us assume that, at a particular instant in time, we experience an anomalous acceleration purely in the  $\hat{x}$ -direction with magnitude  $a_{\hat{x}} = 0.5$  [g]. For simplicity, assume also that there is *no* component of acceleration due to the lurch in either the  $\hat{y}$  or  $\hat{z}$ -directions, that is:  $a_{\hat{y}} = a_{\hat{z}} = 0$  &  $a_{\hat{z}}^{\text{tot}} = g$ . *Without* removing the acceleration due to gravity, we will calculate a net-acceleration value due to this jolt of:

$$a_{\text{net}} = \sqrt{a_{\hat{x}}^2 + g^2} \approx 1.12 \text{ [g]} \quad (12)$$

Note that the 0.5 [g] acceleration “felt” in the horizontal direction is diluted in the net measurement of around 1 [g]. Had we subtracted 1 from the  $\hat{z}$ -direction data, we would obviously had measured a net-acceleration at this instant of exactly  $a_{\text{net}} = 0.5$  [g], which would have been a significantly more useful piece of data for analysis of the track anomaly, as opposed to the  $\sim 0.1$  [g] of deviation from the mean 1 [g] constantly measured in the presence of gravity. It is for this reason that we subtract-off a factor of 1 [g] from all  $a_{\hat{z}}$  measurements.

## 9 Appendix II: Group Photos



Four DAQ devices aboard an Amtrak train.



Daniel MacLean & Kavya Vaidya During an MTD test-run.

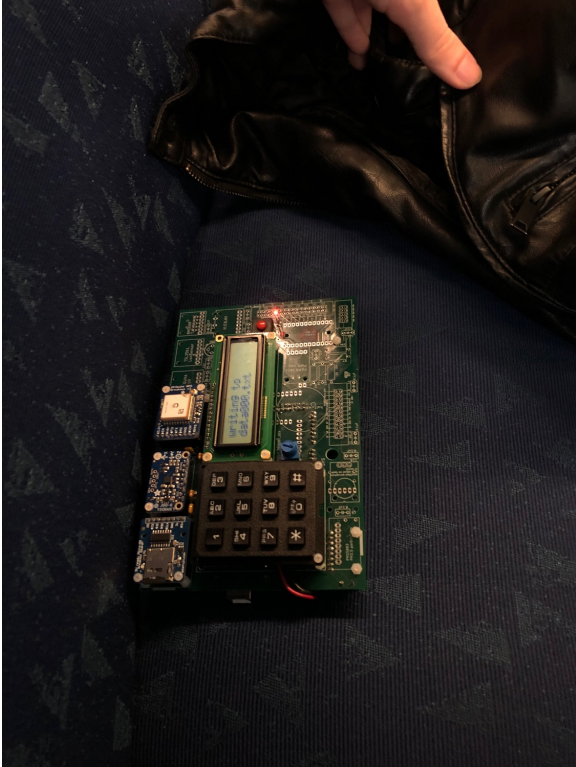


Lauren Gregory, Kevin Na, and Brian Franklin aboard and MTD bus.



Lauren Gregory's DAQ positioned on the floor of an Italian intercity train.

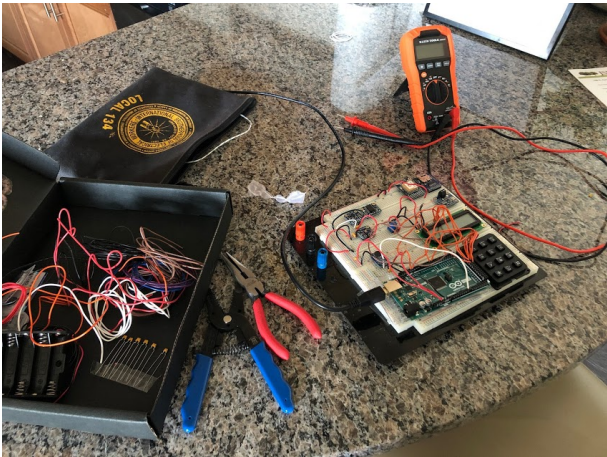




Daniel MacLean's DAQ positioned on a seat during an Amtrak ride.



Lauren Gregory at Chicago Union Station.



Early stages of DAQ design.



Daniel MacLean hoping his DAQ does not become a projectile.

# 10 Appendix III: Amtrak System Map

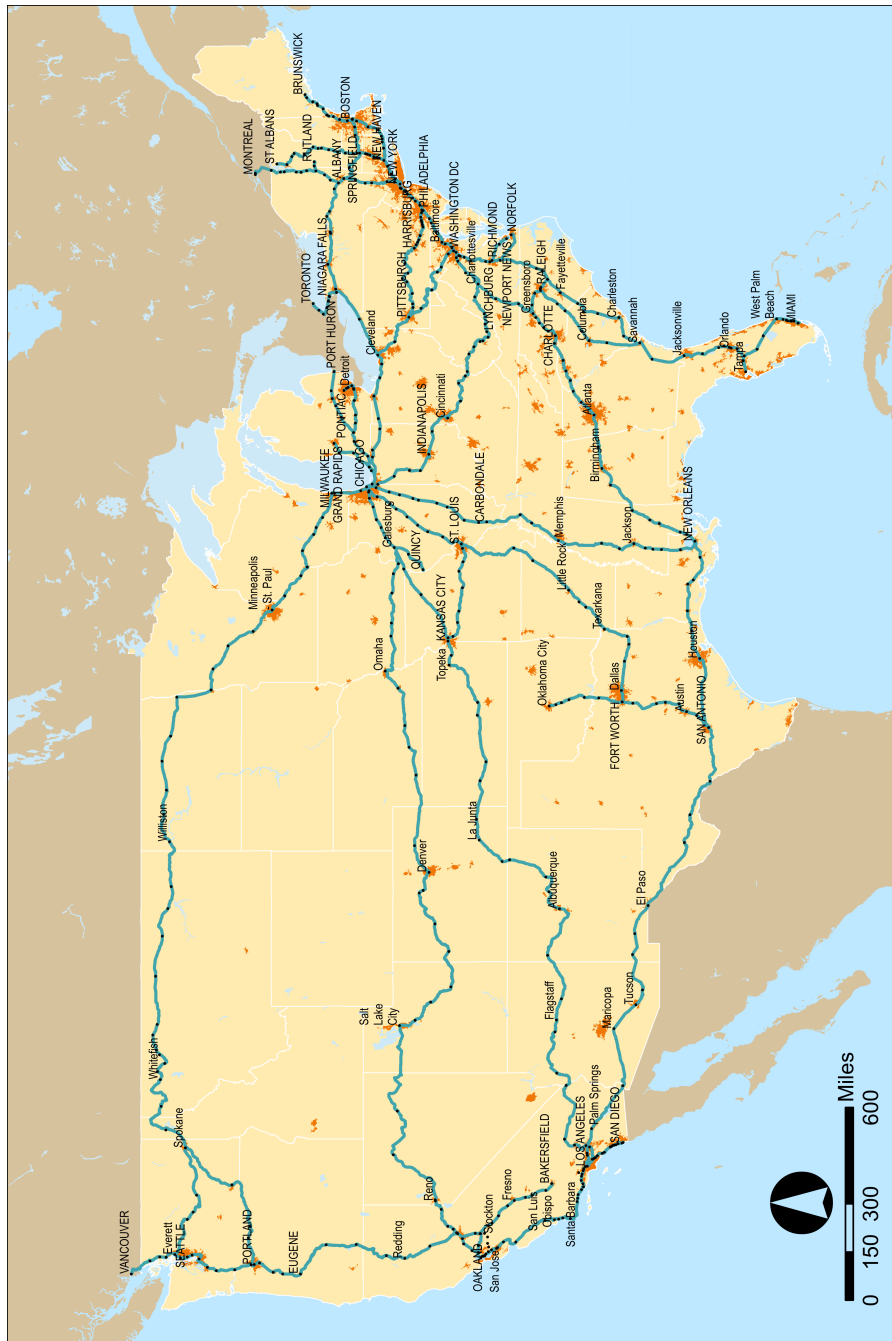


Figure 22: Map of all Amtrak Railways in the United States. Courtesy of Wikipedia; *National Railroad Passenger Corporation*. Train Squad Out.