LMC 6650 Project Studio Seeing Like A Bike

SURFACE TEAM:

Shruti Dalvi

Kristin Hare

Nene Igietseme

Jayanth Mohana Krishna

INTRODUCTION

items used in this project

Seeing Like a Bike is a Literature, Media, and Communications (LMC) project studio that was offered to create space for students to design, engineer and prototype a sensor and computational platform that can sense the physical and social environment of a bicycle. The class was divided into three groups: road surface quality, air quality, and proximity for the researching and prototyping. The expectation is that the different data collection mechanisms would eventually coordinate and complement each other. This design book details the process that the surface team underwent to design a sensor that could measure road surface quality.

PROCESS

methods and steps

- Preparation
- Matrix One
- Matrix One Casing
- Open Computer Vision
- Preliminary Data Collection and Analysis
- Casing for all Sensors

THE PLAN

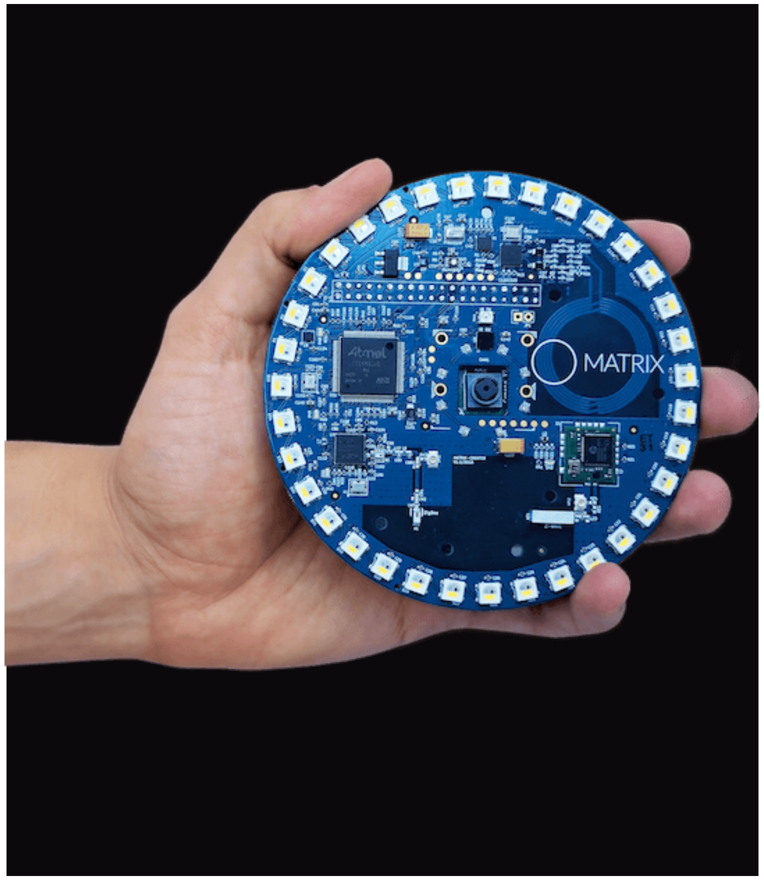
initial elements we considered measuring

What to Measure	Why We're Measuring It	How (Sensor Type)
Swerve (rate of deflection)	**Recorrecting course to show road obstacles and potholes**	**Magnetometer / Gyroscope**
Vibration	**Surface quality (type of surface)**	**Vibration sensor / Contact microphone**
Location	**Location of data collected**	**GPS chip**
Directional sound	Ambient noise, surface quality	Array microphones
Incline	**Slope of road**	**Inclinometer**
Pressure on handlebars	Physiological measure of stress	Area pressure sensor
Temperature and humidity	**Could be related to road conditions and also used to crosscheck primary data**	**Temperature and humidity sensor**

** rows in bold signify that we implemented those elements*

SHOPPING LIST

items used in this project

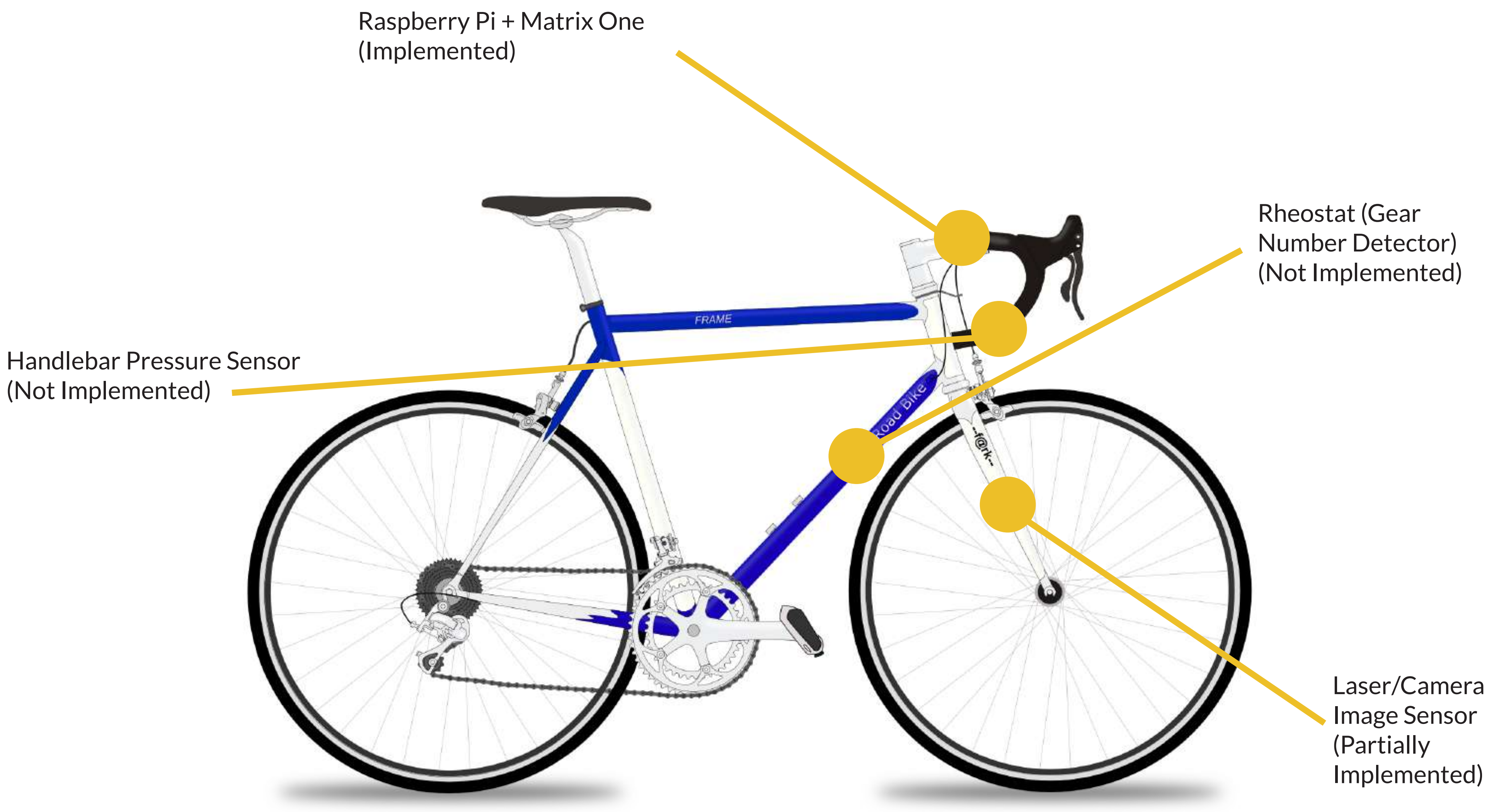


https://creator.matrix.one

Matrix One Creator Edition ($99)
Raspberry Pi 3 ($35)
Anker 26800 mAh portable power bank
Knowles BU21771 contact microphone
GPS chip (in collaboration with the air quality team)
3D Printed Casing

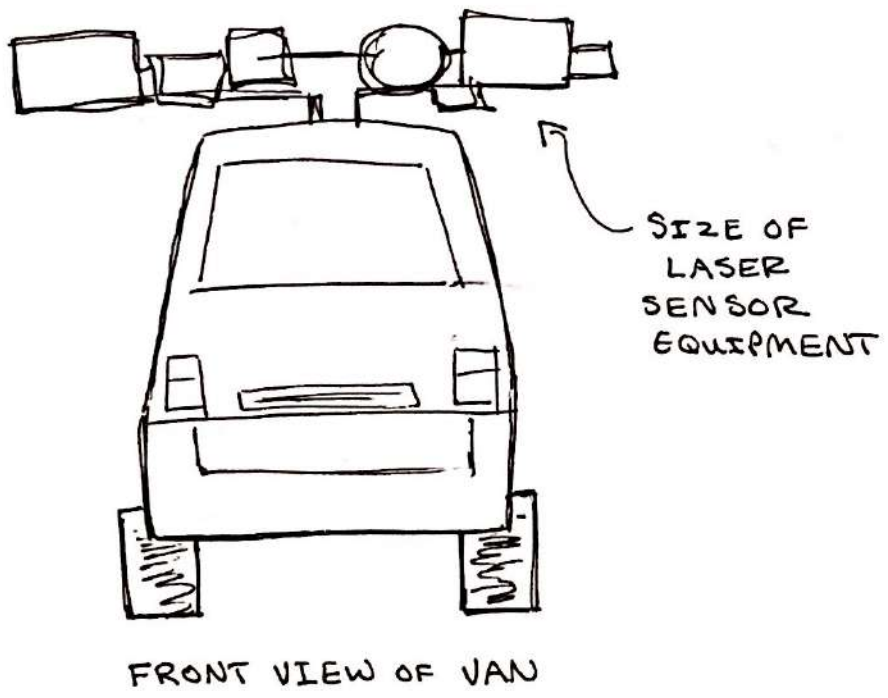
SENSOR LOCATION

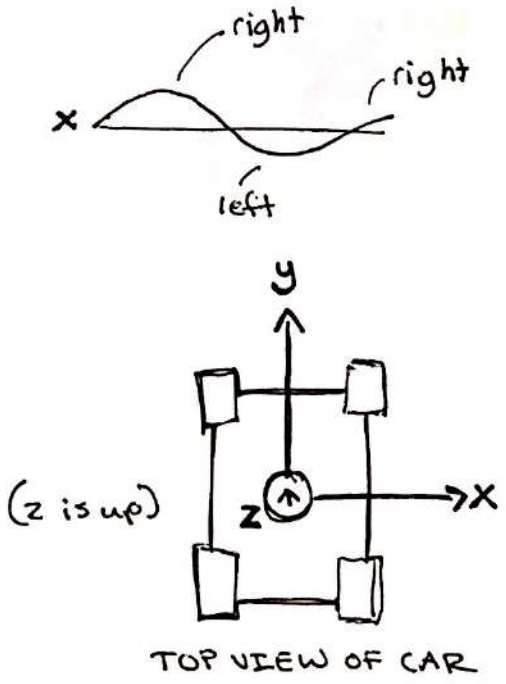
options considered for sensor placement



CIVIL ENGINEERING DISCUSSION

meeting with Dr. Kari Watkins' student to learn how they record roadways



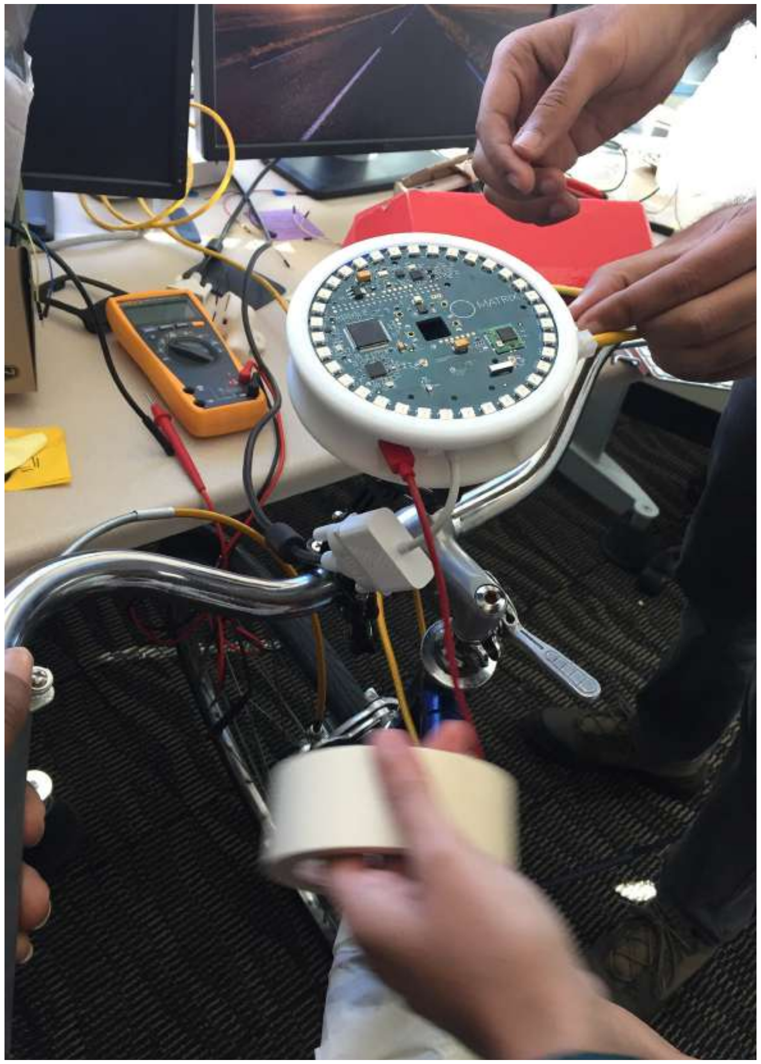


- They use a laser to measure the distance from the laser to the pavement. The laser is not affected by light or reflection (like a camera would be).
- Cost (~$100,000) & size are not feasible for our project

- Student suggested: to calculate magnitude of deviation in data/ road bumps: (x2 + y2 + z2) = t2
- Do not use square root because powerful calculations needed would be drain on the processor.

MATRIX ONE

primary method for data collection



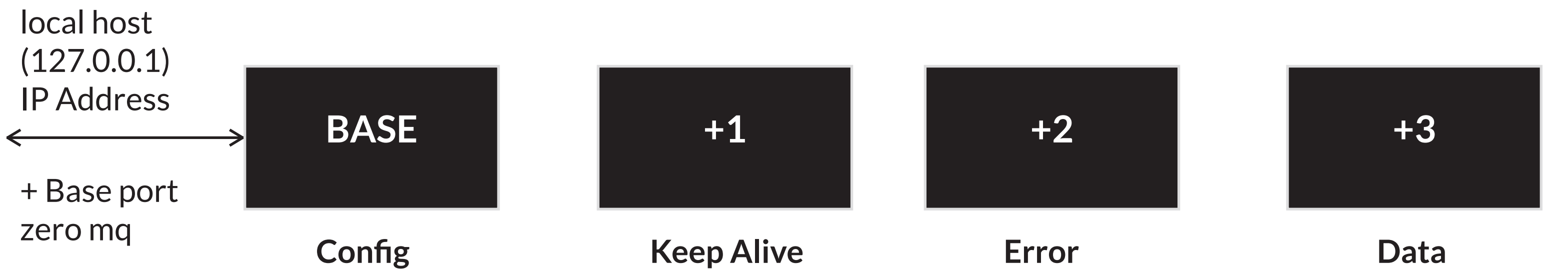
The matrix one is designed to work with a Raspberry Pi. It consists of multiple sensors including a 3D accelerometer, 3D gyroscope, 3D magnetometer, humidity sensor, temperature sensor, ultraviolet sensor, pressure sensors and more.

The Matrix one was placed in a 3D printed case (details on page 10-12), and mounted at the exact centre of the handle of the bike to accurately measure the data. This was the ideal position to measure swerves and to pick up minor deflections in the handlebar in real-time. The other sensors are peripheral, enabling correlation with other data collected.

Refer to the GitHub repository for the code:
https://github.com/cledantec/Cycle-Atlanta-SLaB/tree/master/Surface%20Quality/client%20code

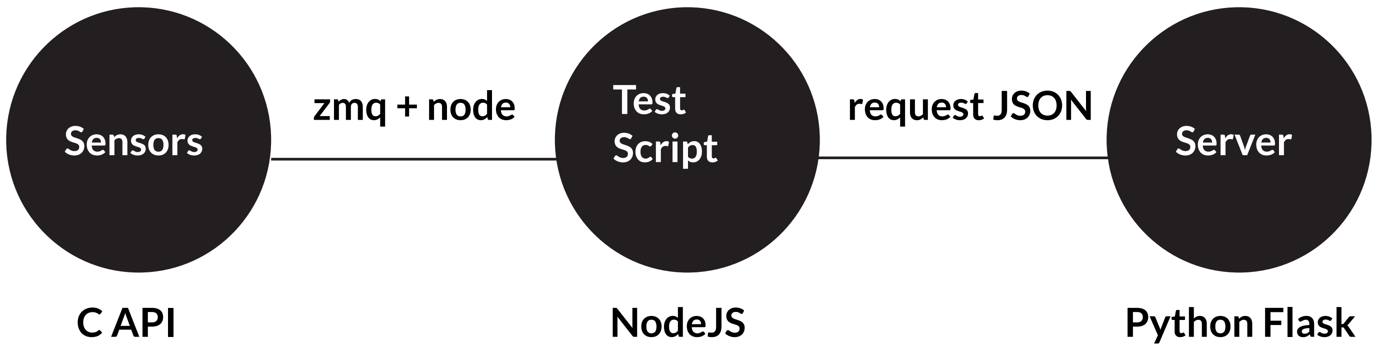
DATA FLOW

from sensor to server



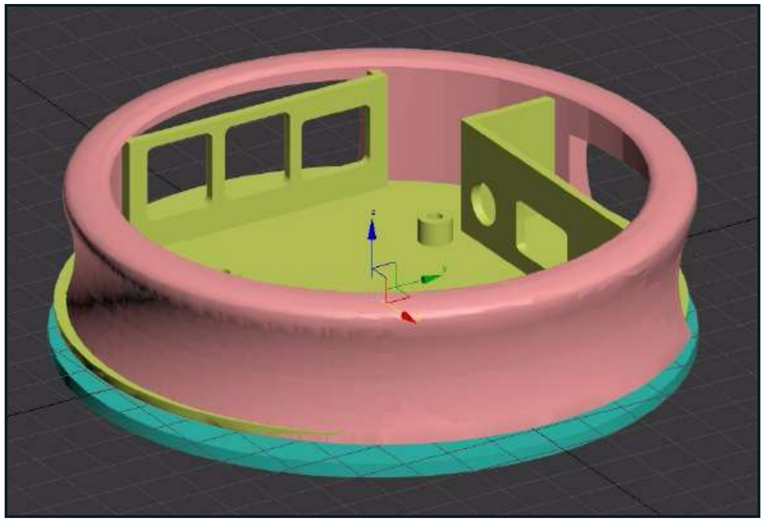
Each sensor on the Matrix has a mapped zero mq port with subports indicated above.

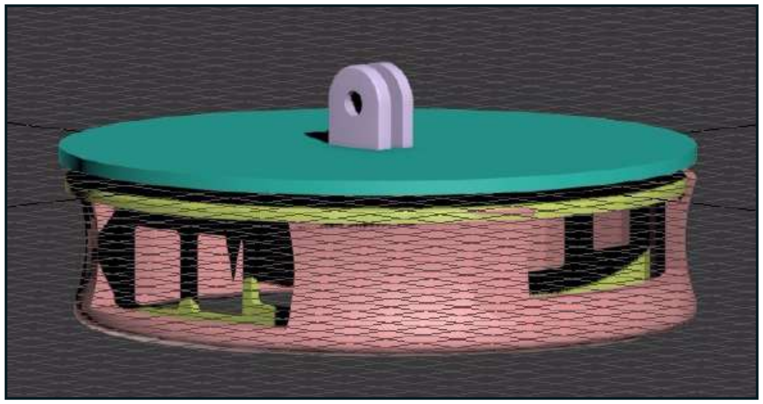
The base port accepts the configuration.
Base +1 keeps the sensor alive by pinging it regularly
Any errors logged by sensor are logged in base+2
Sensor data is logged in base+3



3D-PRINTED CASE

casing holds Matrix and includes a mount for it to be attached to a bike

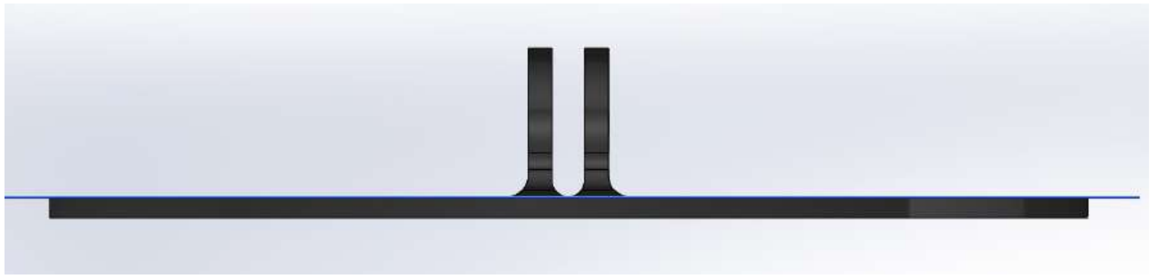




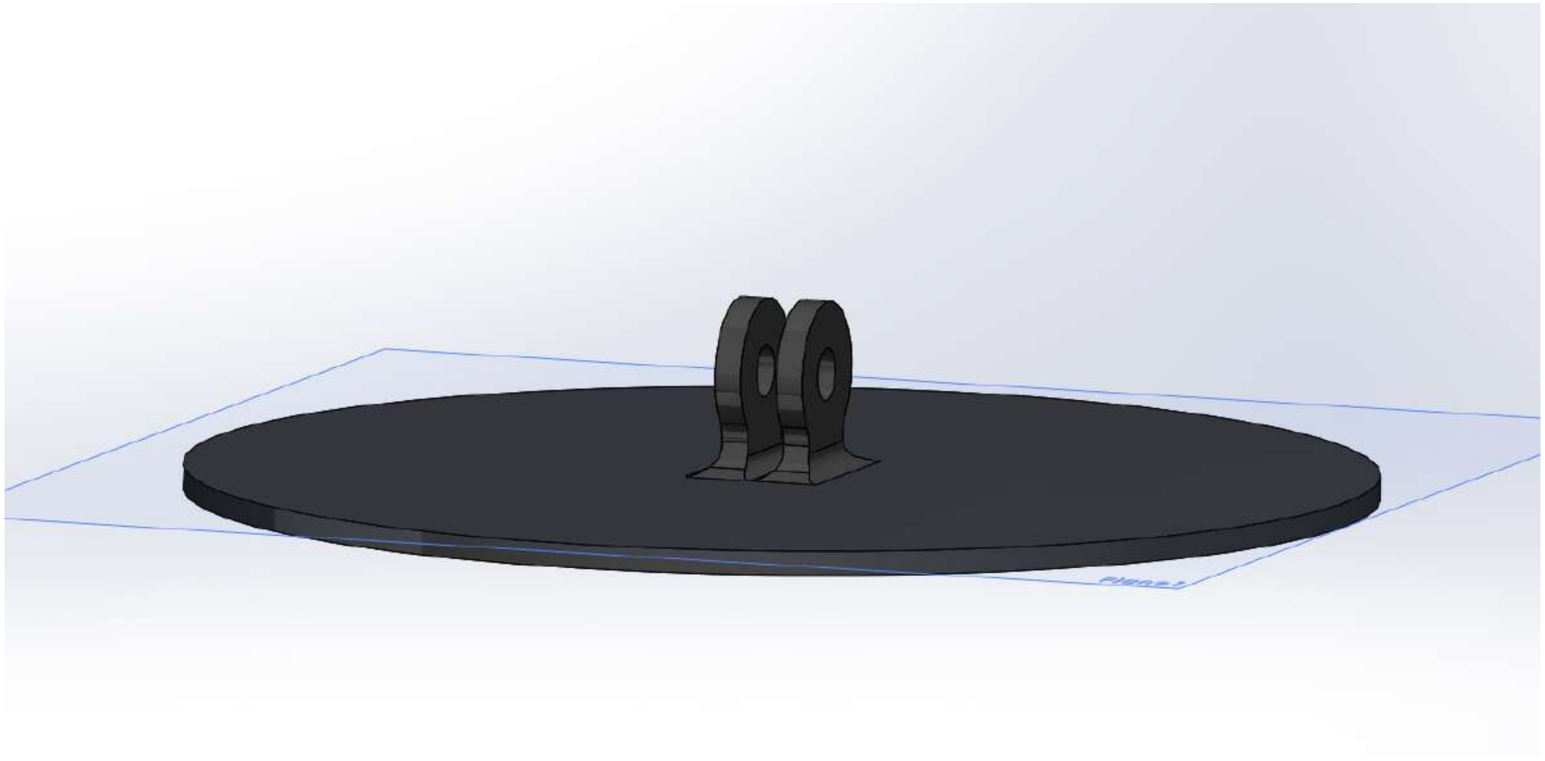
- Used files from GitHub *(https://github.com/matrix-io/MATRIX-Creator-Case)*
- 3D printed at TSRB Prototyping Lab
- First iteration of GoPro Mount pieces broke. 3D printing works in layers and we printed it in a way to save material (i.e., horizontally). This created weak joints where the prongs attach to the disk, which broke in the lye bath. We had to print the part again vertically to get the right joints after modifying them. *Refer to Case Specifications (pg. 12) for details.*

REVISED BASE

added curved supports to GoPro mounts & printed vertically

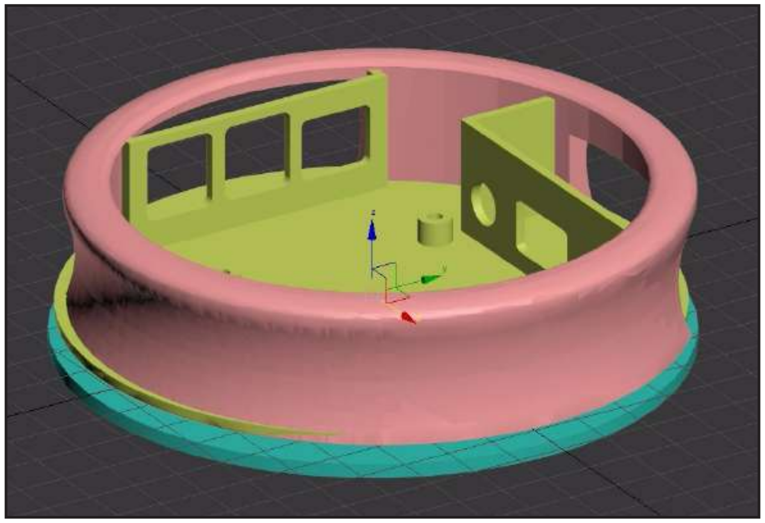


- Base was printed vertically to add strength to the GoPro mount

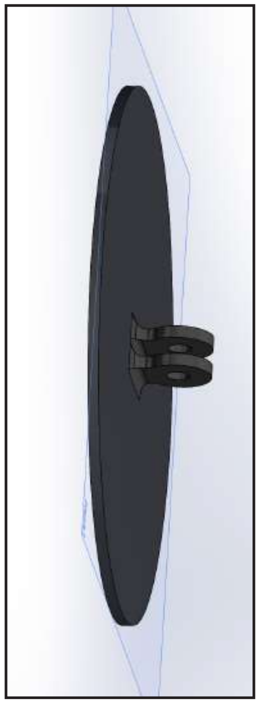


CASE SPECIFICATIONS

steps to ensure proper, sturdy printing



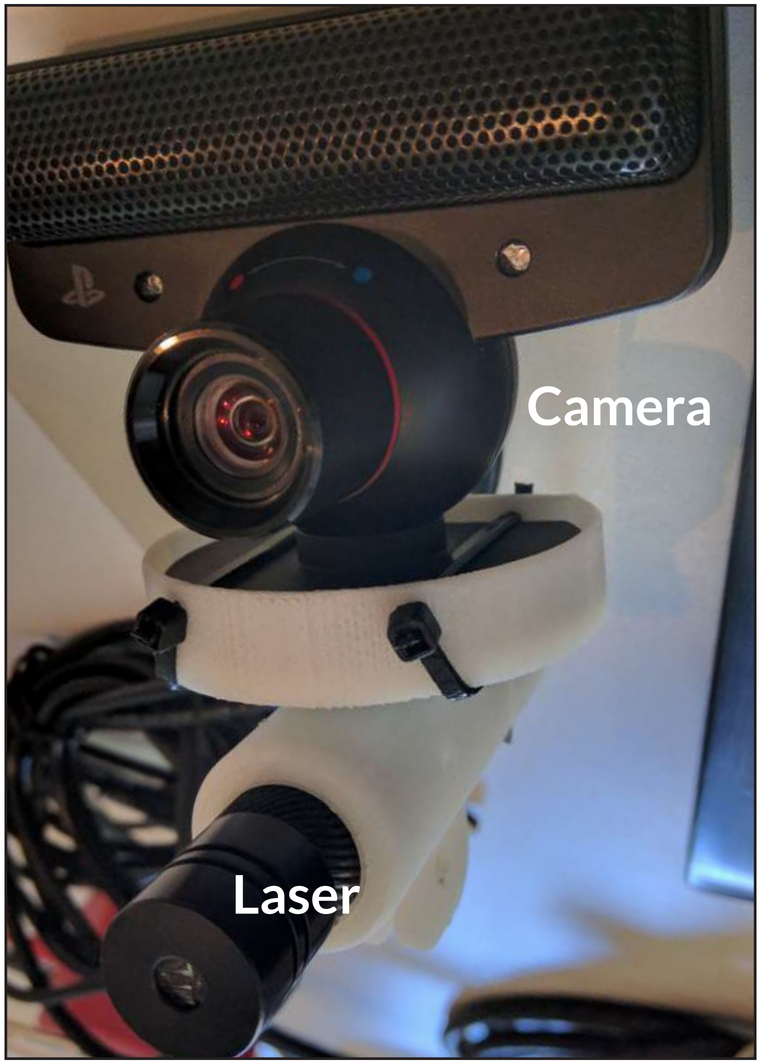
TOP
- Print horizontally
- Settings: Sparse fill, resolution 0.001 in
- Material: white ABS plastic

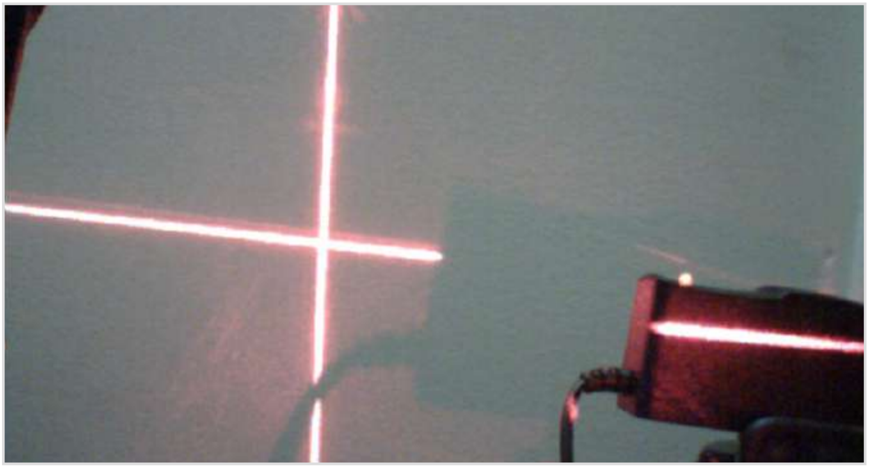


BASE
- Print vertically
- Settings: Solid fill, resolution 0.0013 in (quicker, stronger, but rougher)
- Material: White ABS plastic

OPEN COMPUTER VISION

an alternative method that could be explored further





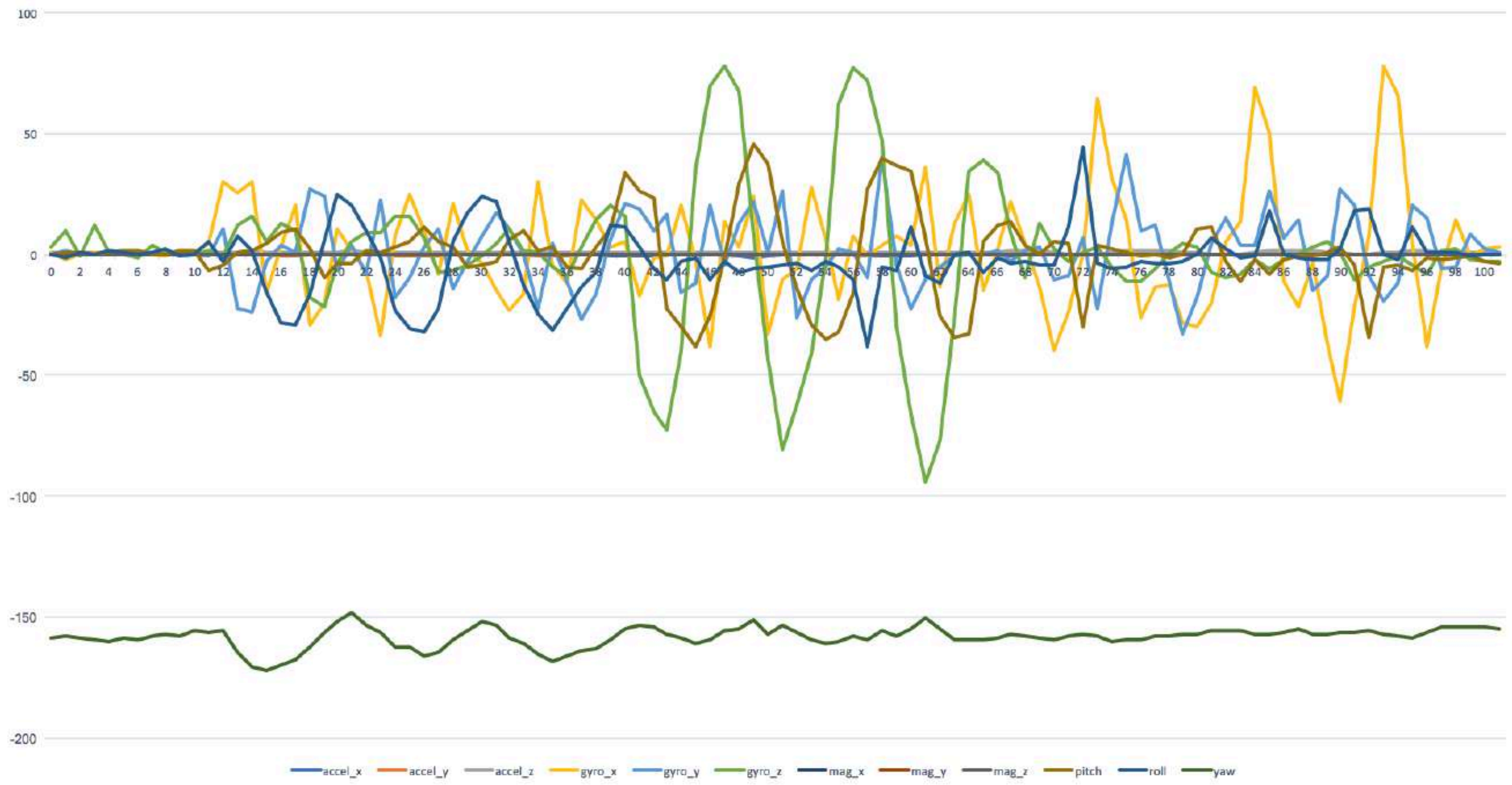
Components used:
- USB camera
- 20mW 3-5V laser diode
- Raspberry Pi

Camera would capture the area on which the crosshair laser is projected. Based on the changes in the laser pattern, the type of surface would be determined. For example, a fine thin line would imply a smooth surface, while a rougher line that seems blur because of distortion may imply a rough surface like gravel, brick or sand.

Preliminary code in the github repository:
https://github.com/cledantec/Cycle-Atlanta-SLaB/tree/master/Surface%20Quality/opencv

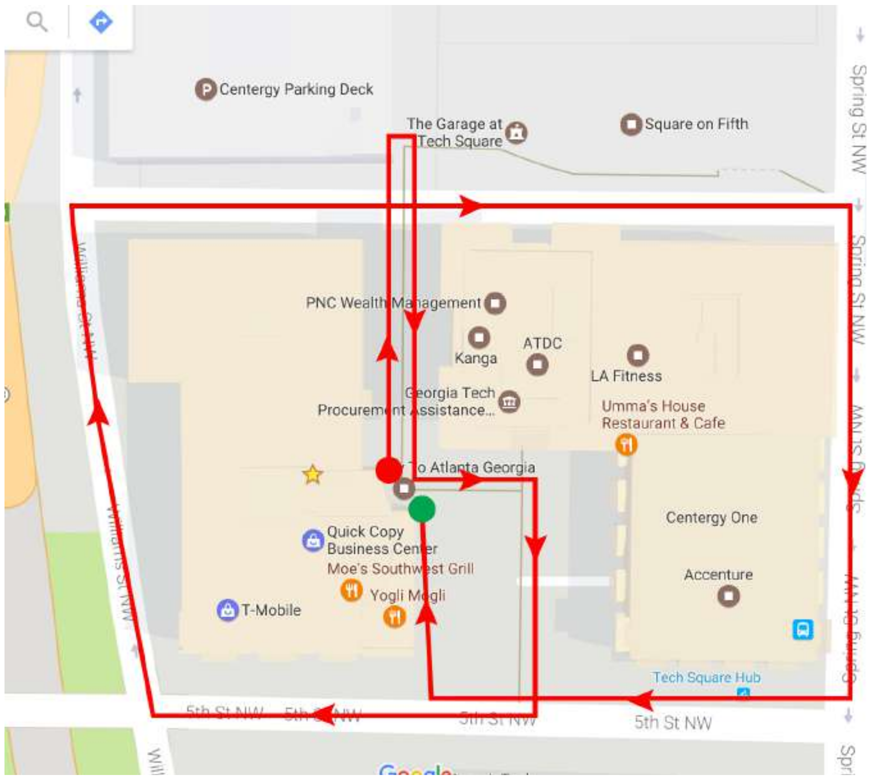
INITIAL DATA COLLECTION

manually shook Matrix to see how data changed



INITIAL DATA COLLECTION

testing route looking at interval levels



We kept the route we took for testing consistent so that we could get data from multiple rides for the exact same location and compare it.

After testing the sensors a few times and through estimation/experimentation we decided the appropriate intervals at which the sensors would collect data.

For the accelerometer/gyroscope/magnetometer data, we determined this interval to be 50ms.

The surface mic collects data every 50 ms.

Temperature, humidity, and ultraviolet rays, and pressure are measured at an interval of 5 seconds.

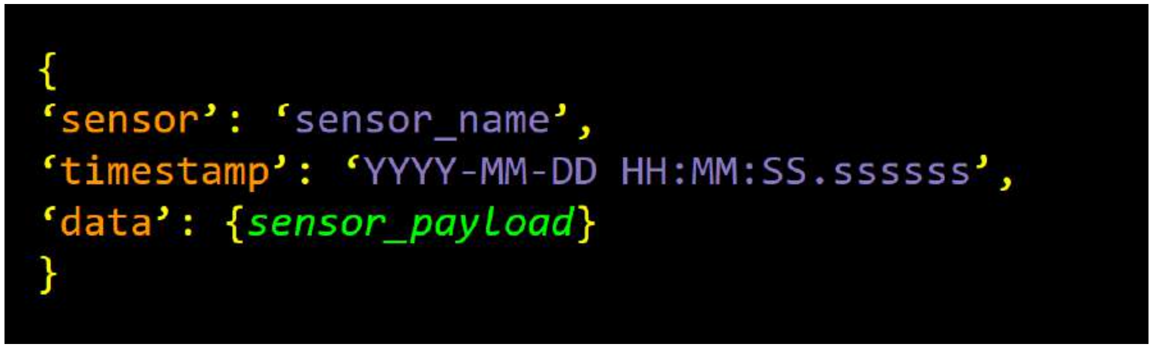
REST SERVICES

data format and location

End point	HTTP Method	Data
getstatus	GET	Returns LED status
status	POST	Gets various sensor statuses
proximity	POST	Gets proximity data
inertial	POST	Gets inertial data
humidity	POST	Gets humidity data
uvindex	POST	Gets ultraviolet data
prestempalt	POST	Gets pressure, temperature, altitude data
gas	POST	Gets gas array data
gps	POST	Gets GPS data

DATA FORMAT

data format and location



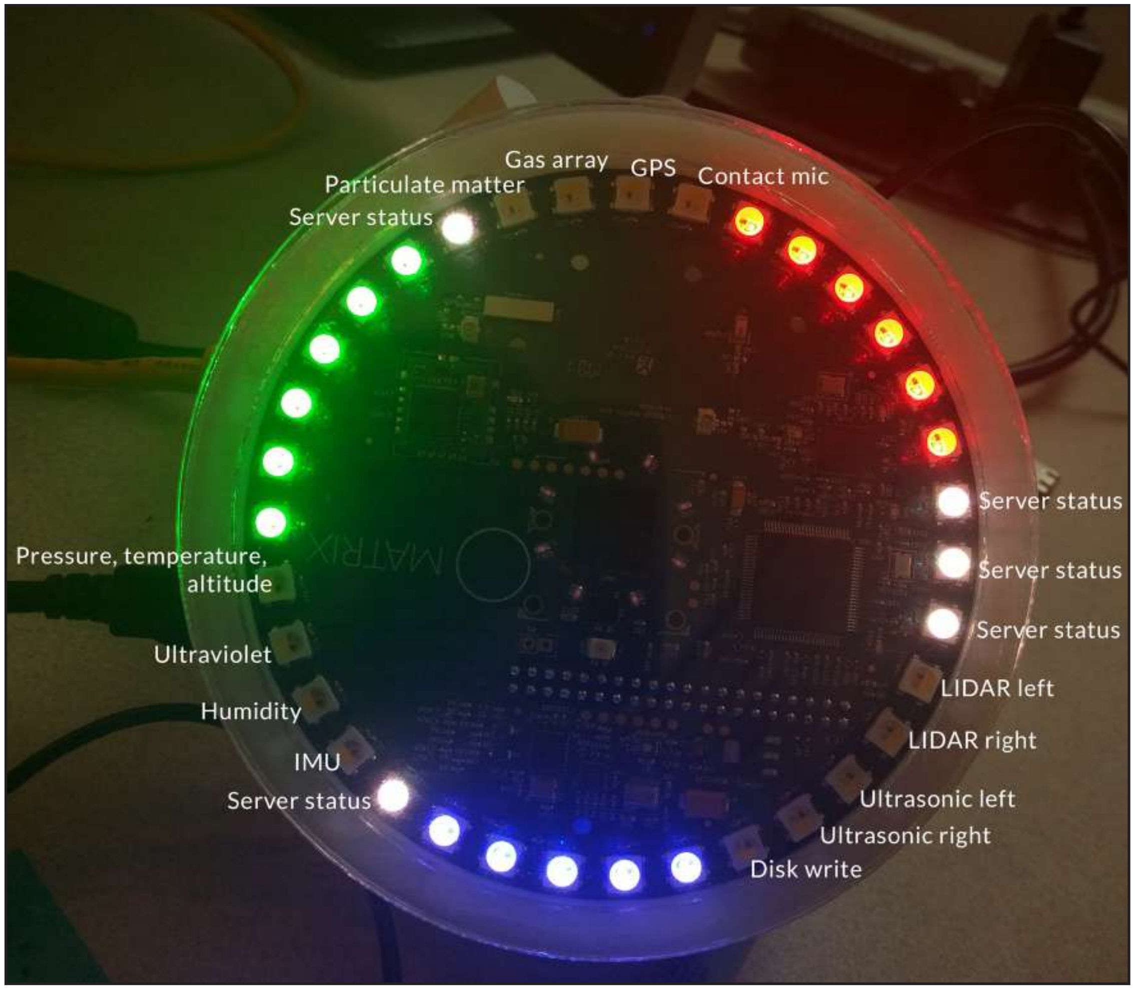
```
{
'sensor': 'sensor_name',
'timestamp': 'YYYY-MM-DD HH:MM:SS.ssssss',
'data': {sensor_payload}
}
```

Data Format

- Sensor_payload is a JSON formatted string that sends server data to one of the endpoints.
- Timestamp is generated on the server when data is received.
- **Data files:**
  imu.log (collects inertial data)
  proximity.log (collects proximity data)
  sense.log (collects everything else)

STATUS LEDS

indicators for different sensors



The LED ring (called Everloop) on the Matrix also serve as a dashboard/visual cue that data is being collected and all the sensors are operational. This is extremely important as most teams faced a problem where they thought the sensors were collecting data while testing but they weren't, either due to some error or that they weren't initialized properly.

Index:
Red: Air Quality
Blue: Proximity
Green: Surface Quality
White: Breathing

INITIAL DATA COLLECTION

data from testing on route

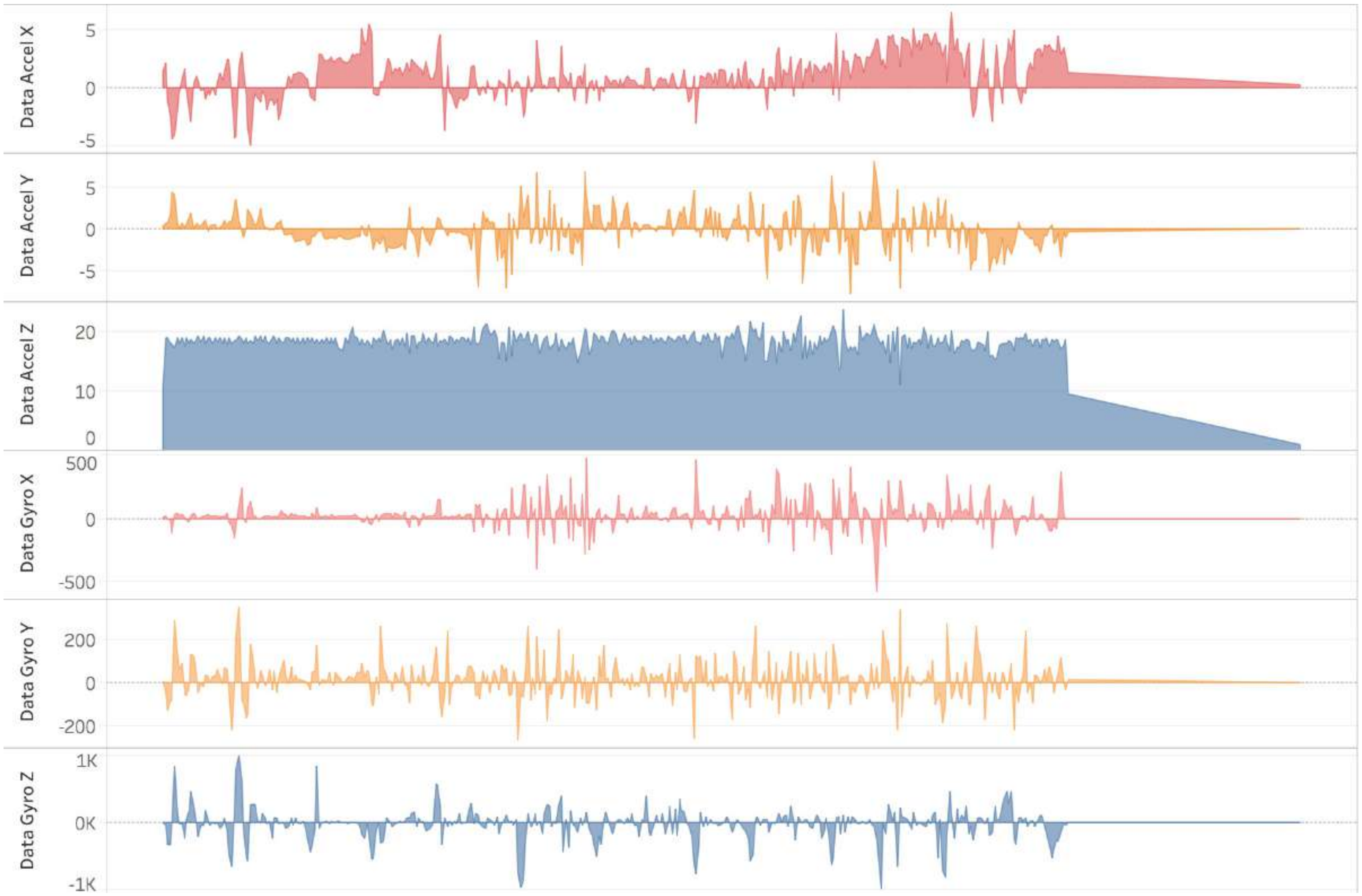


Chart created in Tableau

INITIAL DATA COLLECTION

data from testing on route

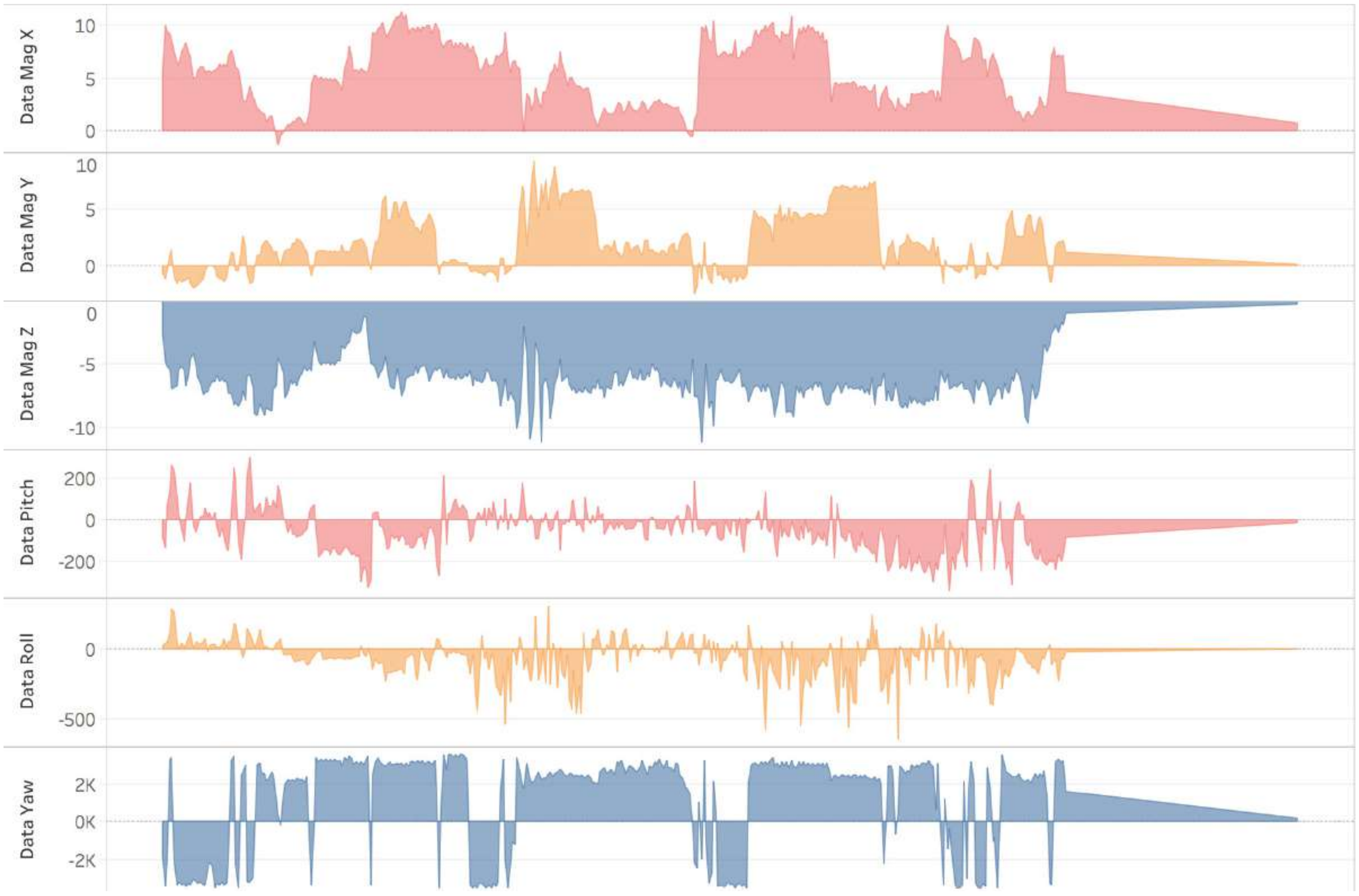
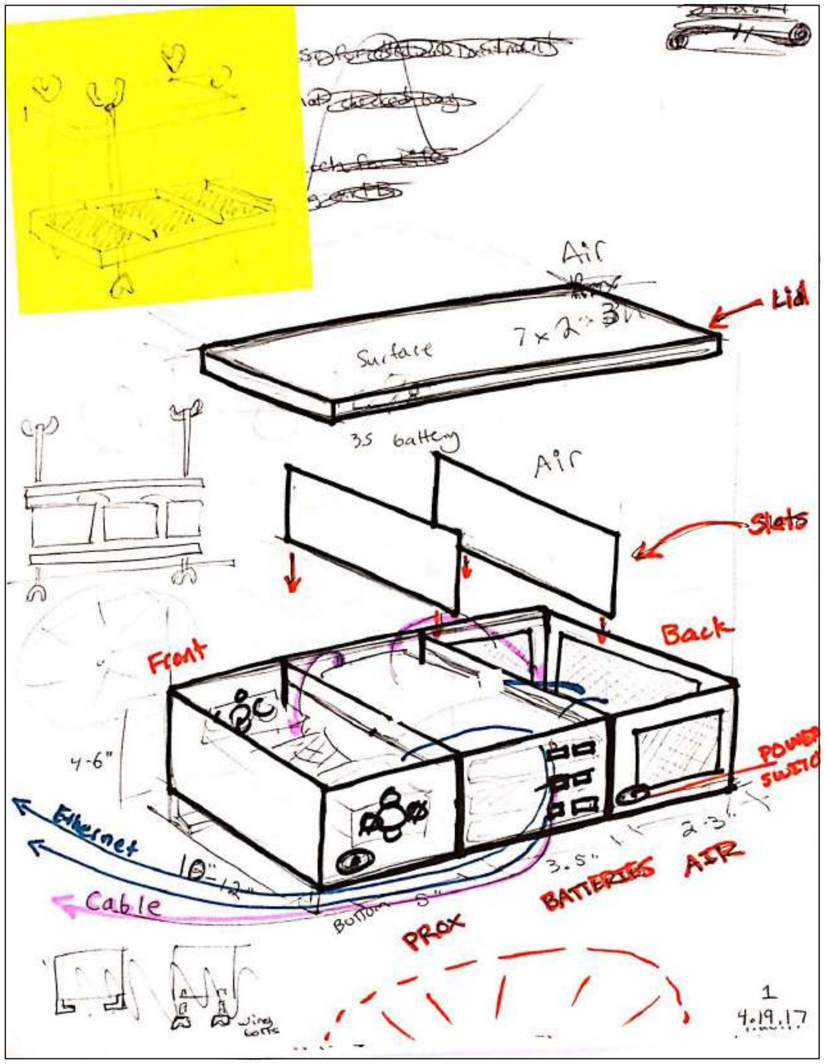
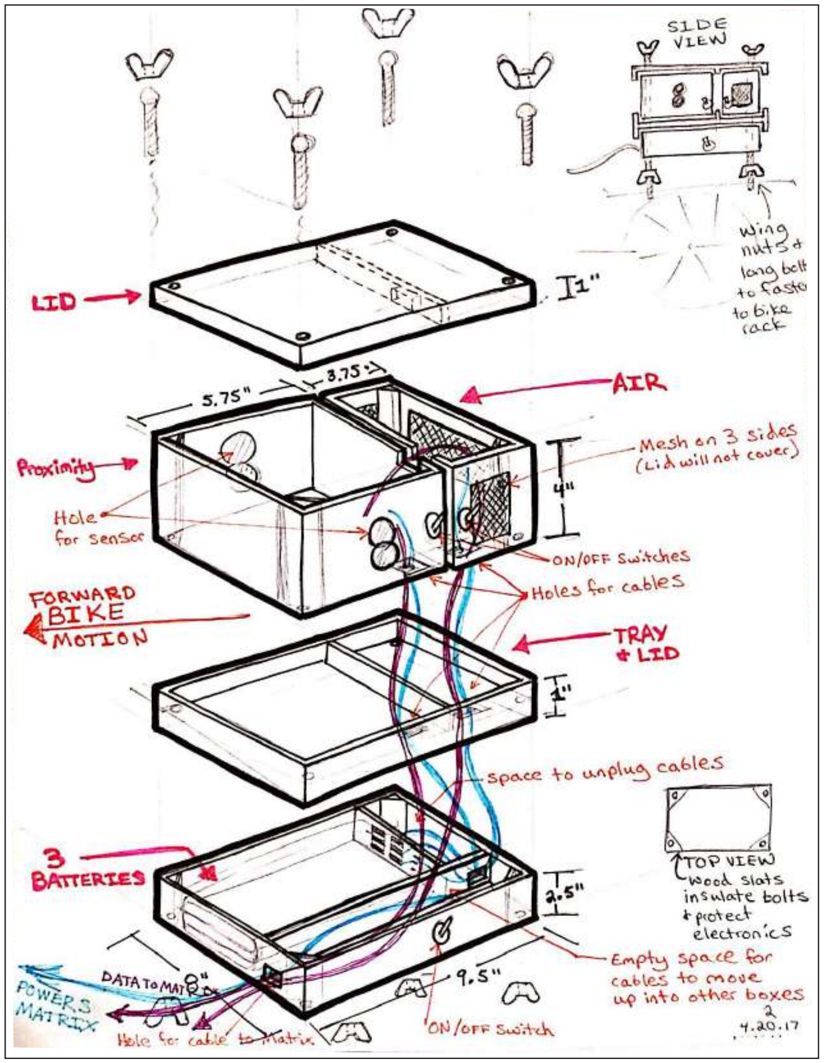


Chart created in Tableau

CASING FOR SENSORS

creating a modular housing for all three team's sensors





NEXT STEPS

pedaling forward & testing the sensors together