Development of a Smart Sensing System for Road Performance Data Collection

Design Document

Sdmay20-32
PROSPER
Halil Ceylan
Ethan Young - Project Manager
Michael Petersen - Hardware
Shlok Singh - Network
Victor Guerra - Software
sdmay20-32@iastate.edu
http://sdmay20-32.sd.ece.iastate.edu

Last Revised: 12/8

Executive Summary

Development Standards & Practices Used

- Kanban Agile development approach
- Data will be transmitted through http requests
- IEEE standard for unit testing
- Real-time operating system (RTOS) for small-scale embedded systems
- Systems and software engineering -- System life cycle processes
- Systems and software engineering — Developing information for users in an agile environment

Summary of Requirements

- Create a on-board device that calculates road roughness (IRI)
- Additional features include taking weather/GPS Information
- Wirelessly communicates data to server/app accessible by phone
- Total cost of device <$100 in order to outfit DoT vehicles

Applicable Courses from Iowa State University Curriculum

- CPRE 288
- CPRE 388
- EE 201
- EE 230
- COMS 227-228
- COMS 319
- PHYS 221

New Skills/Knowledge acquired that was not taught in courses

- Microcontroller programing
- cloud storage
- Sensor data reading
- On-the-fly Calculation on the cloud
- USB readings

Table of Contents

List of figures/tables/symbols/definitions

Definitions

International Roughness Index (IRI) is a standardized measurement of road roughness used by transportation departments nationally and abroad. It is generally described as the amount of vertical variation per unit length of road. IRI is measured typically in meters per kilometer, where acceptable values vary based on speed travelled by the measuring device.

Program for Sustainable Pavement Engineering & Research (PROSPER) is the organization at Iowa State responsible for proposal of the project and support for the design team.

The Iowa Department of Transportation (Iowa DOT) is the transportation authority working with PROSPER to develop and utilize the device outlined in this report.

Symbols

m_s - sprung mass

m_u - unsprung mass

k_s - suspension spring constant

k_t - tire spring constant

c_s - suspension damping

a_v - vertical acceleration

Figures

Appendix B, Figure 2: Gantt chart for project timeline, first semester

1 Introduction

1.1 Acknowledgement

This project is being completed by Iowa State University Senior Design group SDMay20-32 in collaboration with the Program for Sustainable Pavement Engineering & Research (PROSPER) at the Institute for Transportation, Department of Civil, Construction and Environmental Engineering (CCEE), Department of Electrical and Computer Engineering (ECpE) at Iowa State University.

Iowa State University and Dr. Halil Ceyan will be generously donating the financial aid required for this project.

1.2 Problem and Project Statement

The Department of Transportation wants to take regular IRI readings of the many roads of Iowa to identify excessively rough stretches of road. An inexpensive on-board device that can be outfitted on DoT's fleet of vehicles that works with minimal human input. The data must also be stored on a webapp, or remote server, and accessible by mobile phone.

Our task for this project is to create a device that the Iowa DoT can outfit their fleet of vehicles with to take continuous data, throughout the year, on the roads within Iowa. In addition to this, our device can be used to take weather data, and other information while in-use.

The finished project will be a 'vehicle black box' that collects data on road roughness at any given location and stores it on a remote server where the IRI is calculated. The web server will also have a visualization of IRI readings across different roads.

1.3 Operational Environment

Although the black box will be stored inside the vehicle, the device and the sensors within are required to work in the summer and winter. We will be assuming the vehicle has minimal heating and AC to insure the durability of our device. The black box will be fastened/secured, but will be exposed to light dust, and general wear and tear of driving conditions.

1.4 Requirements

Functional Requirements

- International Road Index calculations
- GPS readings
- Web server with automatic data transfer via cell signal (GPRS)

Financial Requirements

- Minimize cost as much as possible.
- No maximum cost requirement to specifically follow, but device should be affordable for a fleet of vehicles.

Environmental/Durability Requirements

- Weather resistant
- Shock proof since the black box will be on poor roads
- Low maintenance

Electrical Requirements

- Use power from the car to provide between 7 and 12 volts to the microcontroller

1.5 Intended Users and Uses

The end users of the black box component will be any person operating a car in the Iowa DOT fleet. These users will not need to use the black box system as it will automatically upload the data to the cloud server. There will also be USB integration for smartphones that will allow for testing. This will be helpful to our team as well as any other team that may use our project at the Iowa DOT. There will also be backups on a USB hard drive/ SD card which can be used to get raw data off of the black box in the event a software failure occurs.

1.6 Assumptions and Limitations

Assumptions

- Maximum price should be as low as possible
- Web server, only a few users viewing data, and only a few devices reporting data

The web server is expected to service a fleet of vehicles, but for the purpose of our project, a smaller scale is more practical to start out with. Future scaling of the server is expected, and on-board hardware is capable of such scaling.

Limitations

- Sensors must be contained inside the black box
- Minimal human-input to start device
- On-board battery powered

Limitations are tied to design constraints. One obstacle that needs to be overcome is achieving accurate IRI readings while keeping the sensors within the box. Further research is needed to find ways to minimize error.

1.7 Expected End Product and Deliverables

This project one physical and one software deliverable for the end user. The physical deliverable is the vehicle black box, which is a housing for a microcontroller and sensors constantly taking IRI measurements and collecting GPS location. The black box will store data using two methods: wireless communication to send data to a cloud server and locally on a microSD card.

The data stored on the cloud converts automatically into a user interface built on a map API showing colored regions of road corresponding to IRI values for that region. It will be stored on a Microsoft Azure server and use the open source OpenLayers API to plot data.

The data stored locally on the black box shouldn't need to be accessed unless there's a lapse in wireless coverage. To access it, the user will simply plug the USB-B port of the black box into a computer and view data saved in text files as a removable media device. The data can be copied onto a local machine and uploaded to the Azure server to process and produce a corresponding map.

2. Specifications and Analysis

2.1 Proposed Design

To take IRI data of roads, various data is needed to conduct these calculations. This includes coordinates from a GPS unit to determine the location of the vehicle, and accelerometers to record the up and down motion of the road. A microcontroller that supports both of these sensors would be a suitable solution that would fit inside an enclosure within the vehicle. With the addition of a GSM unit to the microcontroller, data taken from can be transmitted to a web server for further calculations and storage. A concept diagram for the black box is given as Figure 2.1-1 below.

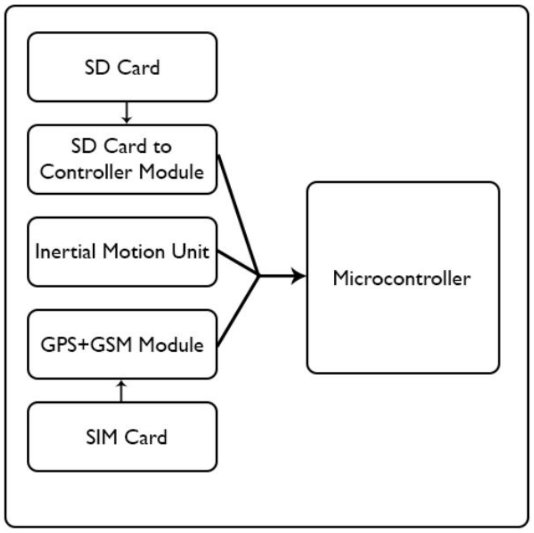


Figure 2.1-1: Black Box Concept Diagram

This method of wireless data retrieval allows us to create a black box that can exist within a vehicle that doesn't rely on user input, or effect the driver or passengers. The web server also allows the calculations of IRI to be done remotely, removing a large form of processing power from the microcontroller. This allows the black box to function with lower power and can minimize the risks associated with the device being exposed to intense climates inside the vehicle.

In the planning phase of our project, research on available microcontrollers and sensors was conducted to determine the hardware requirements for the intended features of our device. The microcontroller must have a sufficient number of input/output (I/O) pins to support all the features of our black box, as well as the appropriate power capabilities for our sensors. Arduino microcontrollers support a large catalog of independent sensors called Shields. This is useful in outfitting our black box with only the sensors necessary to our design, and to avoid spending on components that would otherwise be unused. The Arduino Uno was selected because it matched these requirements without offering more processing power than is needed, or pins that it wouldn't be used. The Uno also has a large number of supported libraries while similar microcontrollers were intended for more niche uses.

IRI can be calculated using multiple different methods. The quarter-car model calculates IRI by positioning two accelerometers set up to the vehicle's suspension. Other methods are similar, but involve placing sensors over all wheels, or both sides. Another method of IRI calculation utilizes laser sensors to measure the distance from the car to the pavement to take extremely accurate measurements of the roughness, but costs significantly more money than its counterparts. In our calculation of IRI, all sensors are limited to the enclosure of the black box inside the car.

Instead of considering the suspension of each car, we will be using the quarter car model with "golden car" suspension, where the values representing sprung mass m_s, unsprung mass m_u, suspension spring constant k_s, tire spring constant k_t, and suspension damping c_s are set to constant values. The constant values must be such that: m_u/m_s is 0.15; k_s/m_s is 63.3; c_s/m_s is 6.0, and k_t/m_s *is 653*. The black box's goal is widespread data of many roads instead of overly accurate data on only a few roads, so the golden-car is close enough to the average car that it will be an easy and accurate estimator.

For the purposes of this project, the accelerometer will be oriented so that the Z-axis is the only axis which records relevant acceleration in the car, since vertical acceleration is an indicator of road roughness (Yuchuan, 1). The conversion from vertical acceleration to IRI, generally, is:

$$\text{IRI} = \frac{\iint_{t_{start}}^{t_{stop}} |\alpha_v|(dt)^2}{S}$$

Equation 2.1-1: IRI Calculation through Vertical Acceleration

where S represents the distance travelled over the time interval from t_{start} to t_{stop} and a_v is the average vertical acceleration over the time period. The goal is to make the time periods minimal and the speed constant across the time period. Double integration of acceleration across the interval will return the difference in velocity from t_{stop} to t_{start}, so a second integration function will return the average relative vertical position over the time interval. The relationship between each data point is normalized using the golden-car model to find the distance tires travel based on cabin movement, and the change in vertical height is then divided by the distance travelled by the car to convert the value into an IRI ratio representation. In this way, lower vertical acceleration values will return lower IRI values, and acceptable IRI values are dependent upon speed travelled. Logically, a car travelling slower will experience more roughness due to more time for tires to change vertical location. Therefore an acceptable IRI value for a car travelling low speed is higher than an acceptable IRI value for a car travelling higher speed. The exact relationship as defined by IRI standards is available in Appendix B, Figure 2.

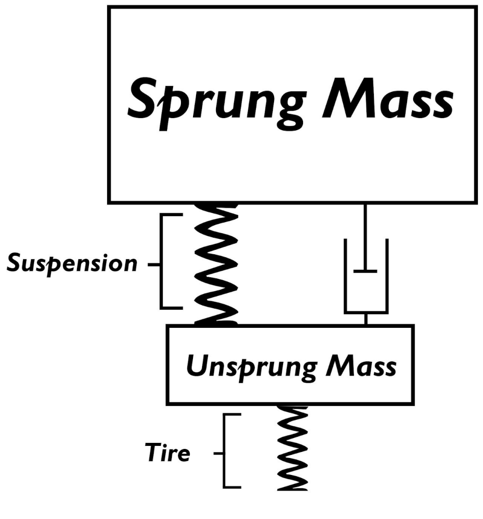


Figure 2.1-2: Quarter-Car Model

The sprung and unsprung masses represent the mass of the car and the mass of the tire, with the springs in between the masses representing suspension, and the spring beneath the unsprung mass representing the stiffness of the tire.

Since IRI is calculated by integrating acceleration data in the sprung mass, an accelerometer must be placed in each vehicle intending to measure road roughness. The microcontroller will allow accelerometer and GPS readings which will be polled relative to the speed of the car, since IRI standards call for polling every 300mm. The raw accelerometer and GPS data will be buffered onto a local removable microSD card, and each sample will be sent over HTTP using the 2G wireless network by the GPRS module to a remote server.

The server will run as an Apache web server, using C code to receive the raw data from the black box. The code will be parsed into a SQL table, then converted to IRI using Javascript. Javascript will be a lightweight way to implement difficult calculations, and is typically compatible with web servers. The parsed data will be placed into another SQL table. A concept diagram of the design is given as Figure 2.1-3 below.

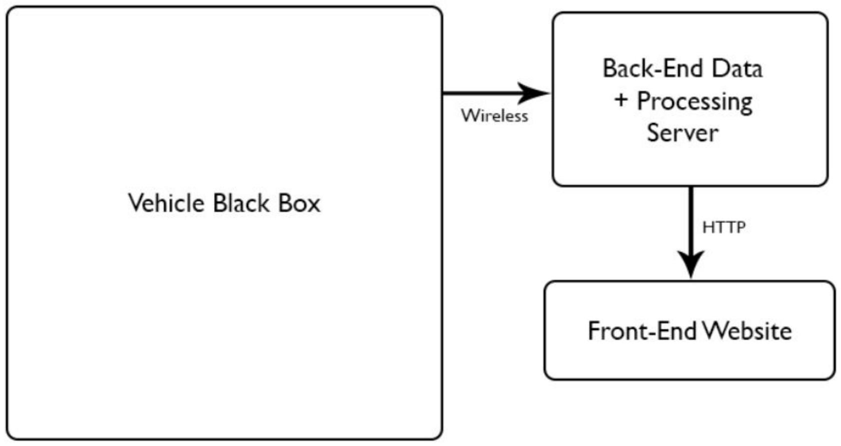


Figure 2.1-3: Concept for Design

The design is meant to be scalable, where many black boxes will interface with the back-end server.

The front-end design, although optimistic, will run on the OpenLayers map API and make requests to the SQL table to retrieve IRI data. The OpenLayers API is free, open source, and feature-rich, making it a good candidate for a front-end display tool. A visual conceptualization of the front-end application is shown below.

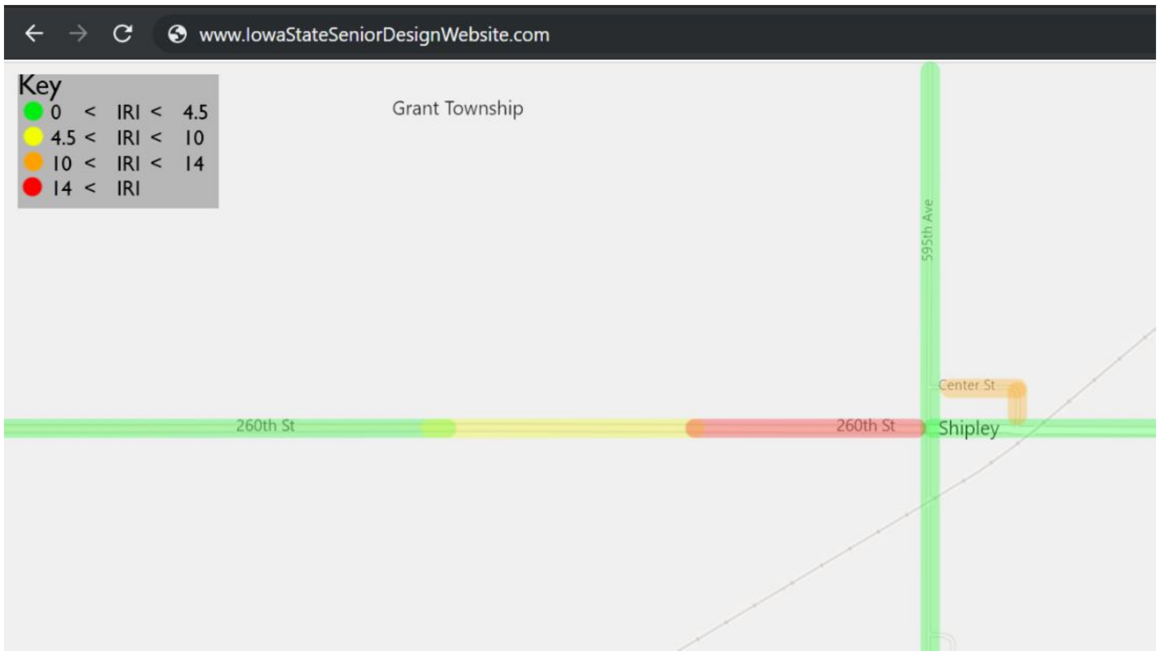


Figure 2.1-4: Front-End Application Visual Concept

The colored regions will be interactive for users, where clicking a region will show information about the time, speed, and vehicle information that was collected alongside the IRI data. Since acceptable IRI ranges vary based on the speed of the vehicle, as faster vehicles are expected to experience more vertical turbulence, mapping the speed alongside the IRI will be crucial.

2.2 Design Analysis

Research into microcontrollers and sensors has allowed us to purchase parts at reasonable cost. IRI calculations were determined using a physics-based approach that uses a quarter-car standard model of IRI calculation. It's necessary to perform some filtering on accelerometer data to reduce noise and drift caused by the electronics. Our initial idea was to use hardware filtering to subtract low and high frequency data, but the implementation proved challenging, and software filtering is cheaper. For software filtering, we will use Fourier transforms to compare acceleration versus frequency, where low and high frequencies will be ignored, working as a bandpass filter. The exact cutoff frequencies will be determined experimentally. In addition to a software implementation of a bandpass filter, a moving average for acceleration values can also be applied to increase the accuracy of accelerometer data.

2.3 Development Process

For this project, we will follow an Agile development process and more specifically a Kanban approach. This will help us to focus on only a few things at once to get them tested and working before trying to tackle new functional requirements. Kanban will be better solution for us rather than the Scrum model because with our busy schedules, we cannot afford to meet as often as the Scrum model entails. We will most likely only be able to meet one or two times per week. GitLab has an Agile board included in its software that we will utilize for this process.

2.4 Design Plan

Transmitted using cellular data, raw data containing timestamps, GPS coordinates, and acceleration values will be uploaded to a web server which will perform data processing to convert the raw data into IRI values for corresponding GPS coordinates. In addition to this, the web server will send the processed data to a client application and use the OpenLayers mapping API to display for user interpretation.

There are 3 main use cases for the black box: installation in vehicles, manually uploading data to the Azure server for processing, and using the client application which displays IRI data across roads.

To address the first use case, installation of the black box, the user will install the device in a predetermined suitable location. This will need to be above one of the four wheels of the vehicles. After connecting power to the black box, GPS and Accelerometer data, the device will begin collecting and transmitting.

The second use case is in case of device failure. In the event of a hardware failure, or data from the black box could not be transmitted, a user can manually retrieve data from the SD card and upload the data to the server. The local data on the SD card will include timestamps, GPS locations, and instantaneous acceleration values. From there, the server will process the data as if it were transmitted.

The third use case is the user interpreting the processed IRI data. The back-end of our web server will automatically determine when the vehicle is driving and suitable IRI calculations can be performed. IRI data will be displayed on the front-end. The most current data will be displayed first, while data from past months can also be displayed. The user will be able to access the web server from any computer, or mobile phone.

The server back-end will process data to calculate IRI values across the available GPS locations, assign them to the nearest roads, and allowing the creation of an interactive map displaying known IRI values on roads to address the third use case of the design.

3. Statement of Work

3.1 Previous Work And Literature

Currently there exists a variety of mobile apps such as *Roadroid* that calculate the IRI of roads using the sensors built-in to a mobile phone (Roadroid, 2019). *Roadroid* also takes photos of the road and displays a map of the recorded data. Although the app shares features that will be included in our device, the medium and methods of taking and transmitting the data will be different. One of the problems with RoadRoid is that the user must own a smartphone that is capable of taking data. When outfitting potentially hundreds of vehicles with a device to calculate IRI, the potential issues of using a smartphone are greater. The expectation of someone using their personal phone to collect data is unfeasible in this scenario and providing smartphones for all vehicles will be too expensive. Another added benefit of a black box, is that once it is installed in the vehicle, no other input is needed to continue data collection or transmission.

A significant amount of research into IRI calculation and we have found that the double integral method to be the one we want to use. The double integral method outlined in Equation 1 uses an accelerometer to get the vertical acceleration while driving. This will require a band pass filter for the initial acceleration data, then another for after the first integration, and one more for the final integration. This is because the integration produces drift every time. In the beginning there will also be noise produced.

Hardware components were selected carefully. To calculate vertical acceleration for IRI calculations, an accelerometer that can withstand g-forces that are reasonable for a bumpy drive is needed. An 8-g accelerometer was chosen because a car should never accelerate in any direction faster than 8 gs, and it is low enough to maintain accurate measurements for lower values. Digital accelerometers are more popular and sparkfun.com states that they "are less susceptible to noise" (sparkfun.com, 2019). As for GSM unit, the main factor in our decision was the price point. Arduino supports multiple GSM modules, however, the newer models that support 4G communication were too expensive for our targeted price point, and unnecessary for our application. The Fona 808 supports 2G communication, and also includes a built-in GPS unit. At a lower price, this module was an obvious choice for our device.

3.2 Technology Considerations

The black box will be built on some microcontroller with modular sensors. Because of the budgetary requirements of the design, the cheapest reasonable microcontroller was considered. Arduino systems were compared with Raspberry Pi systems. Although the Raspberry Pi is more full featured in terms of memory and compiler ability, various versions of the Arduino are cheaper with sufficient functionality.

In terms of accelerometers, the weakness of analog output accelerometers is that they produce more noise than their digital counterparts. Throughout investigation of accelerometers available, it was also evident that many of the digital accelerometers also use the SDA pin on the Arduino. This makes it very easy to hookup and use as there are many libraries available for using an accelerometer in this way.

The GPS/GSM module was chosen for its cost and functionality. With a resolution of 2.5 meters for GPS readings and 2G support, it meets our requirements for data collection and transmission. 2.5 meter resolution means it will be easy to correlate coordinates with nearby points on roads when processing data.

3.3 Task Decomposition

The tasks breakdown will follow the project timeline. After a period of literature review, the microcontroller will be selected so that each other black box component can be researched and ensured compatible with the controller. During this time of device research, IRI measurement parameters will be known, and code will be built around them. Actual hardware interfacing and embedded design will be accomplished at the end of the first semester of the project once components have arrived.

1) Literature Review
    a) IRI interpretation
    b) IRI theoretical calculation
    c) IRI actual implementation
    d) Component consideration
    e) Software consideration
2) Sensor Configuration
    a) Testing accelerometer and analyzing outputs
    b) Testing GSM/GPS module and analyzing outputs

    c) Testing Arduino interfacing
3) Assembly
    a) Prototyping with breadboards
    b) Re-testing components and interfacing with Arduino
4) Coding
    a) Embedded design
    b) Embedded implementation
    c) Back-end data handling design
    d) Data transport design
    e) Front-end data handling design
    f) Data transport implementation
    g) Back-end data handling implementation
    h) Frontend data handling implementation
5) Testing
    a) Test interfacing with each hardware module versus expected values
    b) Test data transport reliability
    c) Test data transport speed
    d) Test back-end data processing for IRI calculation accuracy
    e) Test back-end data processing for proper GPS coordinate-to-road assignment
    f) Test front-end UI with webdriver
    g) Test front-end with dummy back-end data
    h) Test front-end with real implementation

3.4 Possible Risks And Risk Management

Risks

- **Technical Risk:**
  One or more components might not interact correctly.
- **Cost Risk:**
  Implementation may be relatively cheap for prototype, but for mass production expensive
- **Scheduling Risk:**
  The project might not adhere to the expected milestones or timeline perfectly
- **Sustainability Risk:**
  The 2G network will be disabled at the end of 2020, but the black box relies on 2G for proper function and minimal cost

Risk Management

- **Technical Risk:**
  Make sure the components have reliable and reproducible behavior, and try to minimize issues during prototyping and coding.
- **Cost Risk:**
  Minimize requirements for human interaction, consider costs for mass purchasing, document process thoroughly so as to create a easier set-up in each fleet vehicle
- **Scheduling Risk:**
  Attempt to work with contacts to overcome challenges as they come up

- **Sustainability Risk:**
  Attempt to make the GSM device as modular as possible, so that future iterations of the black box can be easily switched to 3G or 4G networks

3.5 Project Proposed Milestones and Evaluation Criteria

- The first milestone will include a fully functional and assembled Arduino. This will include all modules such as the accelerometer, GSM and GPS, and SD card modules working 100%.
- The second milestone will be of a working IRI calculation. Since the web server may not be ready quite yet, the data will be saved to the SD card for inspection. This milestone will include setting automated intervals for retrieving data at each 300ft as well as having code to take that data a retrive the IRI calculation from it. We will use data that has been previously collected by PROSPER to compare our results.
- The third milestone will be to have a web server that the Arduino assembly will be able to contact. The Arduino will need to be tweaked to send data at the automated retrieval points already created. The web server will then take this data and handle the IRI calculations and save them into a database.
- The final milestone will include creating a user interface to interact with the back-end to see IRI calculations on the roads around Iowa. For this, we are using a map API called OpenLayers and we will need to make sure that the points on the map correlate to where the user drove.

3.6 Project Tracking Procedures

We use GitLab to track our work through the semester, uploading code and maintaining problems through their ticketing system. This makes it easier to see what everyone is working on and to access their contributions provided they push code every time they work on it.

3.7 Expected Results and Validation

The desired end product will be a closed system device which sits inside any car and connects via USB to a power source to measure road roughness and transmit IRI data based on GPS location to an off-board storage location.

At the highest level, using the black box on roads of various roughnesses and comparing measurements with alternative measurement devices, such as the Roadroid app and laser truck rig, will give us a good idea of the accuracy of our system.

Back-end data processing validation will be done using dummy data whose IRI values are calculated by hand. This will allow comparison of code accuracy versus real calculations. Each function coded on the back-end will be subject to rigorous unit testing including testing of edge cases.

Frontend processing will be similar, where known dummy data is fed into the code and expected versus actual results are compared. Unit testing will again be rigorous.

Data collected by the laser truck rig will be able to get processed through our back-end code and front-end to compare to our own hardware and validate hardware.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Semester 1	
By 10/14/19	Literature review & background research
By 10/21/19	Microcontroller analysis, System/Code planning
By 11/4/19	On-board sensor research, software/hardware compatibility
By 11/18/19	GPS study, Early code 'rough drafts'
By 11/25/19	GSM research, finalize component list for purchasing
By 12/2/19	Early prototyping, Code Testing
Semester 2	
By 1/13/20 (over break)	Testing individual components
By 1/27/20	Assemble Arduino and functionality test for components
By 2/17/20	Implement data sampling rate
By 2/24/20	Implement internal storage
By 3/2/20	Communicate with web server
By 3/9/20	Database integration
By 4/6/20	Working model with IRI data sent to remote server using GSM + long range testing to simulate real-life conditions
By 4/27/20	User accessible website with easy to view data, and final touches to make device user-friendly

A long initial period of literature review is required to attain familiarity with the project details. The IRI calculation is the crux of the project, and understanding its calculation is crucial to hardware selection. The code will perform the actual calculation of IRI, so having a general idea of the translation of math to code will allow a smooth implementation of hardware. Then, once we know what data our code will need we'll be able to begin selecting parts to produce this data, but selecting the right parts for the best prices will take some time.

We'll select a GPS module as we begin our coding efforts. We'll also select a GSM module and finish up the remaining component purchasing over the following weeks. Very early prototyping should begin by the end of the semester, aiming to obtain a bit of familiarity with the hardware while inspecting our code's interaction with the modules.

4.2 Feasibility Assessment

A successful project will be as follows:

1. The Arduino is capable of sending data to the web server as well as storing data on the device. Both will need to be done in real time. Stalling will only be acceptable when wireless communication services are unreachable; in this case, the Arduino would store the data for later transmission.
2. The web server is scaled to be able to handle potentially hundreds of requests at the same time. We do not want the web server to unexpectedly stop or stall for a significant duration of time.
3. The UI is able to correlate data with roads on the map API from OpenLayers. This will need to be completed to a high accuracy as this will be how Iowa DOT will prioritize road construction.

4.3 Personnel Effort Requirements

Task	Effort Required	Explanation
Literature Review	10hrs/wk, 7+wks	Literature review is the foundation of our project. Understanding requirements and calculations allows us to create a functional project, so substantial time is devoted to getting a thorough understanding of theory
Microcontroller Selection	4hrs/wk, 2wks	Microcontroller selection will take a couple of weeks to complete as it will be the basis of our circuitry. We just need to make sure that the selected microcontroller will have enough pins to connect all of the shields that we will need.
Internal Sensor Selection	4hrs/wk, 4wks	The non-GPS/GSM sensors will need to meet specific requirements for measurement precision, and must also adhere to pricing standards.
GPS+GSM/GPRS Device Selection	2hrs/wk, 3wks	We will select the components required for the project, and start working on implementation of the GSM module, we will also work on the back-end in this portion
Early Prototyping / Tests Generation	10hrs/wk, 2wks	Early prototyping will require a couple weeks of dedicated time outside of testing during our initial device setup. Testing IRI calculations will likely require revision as we improve the accuracy of our device.

Testing individual components	15hrs/wk, 2 wks	Testing of each component will first require getting the microcontroller configured, then wiring all of the components to it one after another.
Assembly and group component testing	15hrs/wk, 2 wks	Assembling all of the components will need to be carefully done looking at all the datasheets provided with the components.
Data sampling rate implementation	15 hrs/wk, 3 wk	Fine tuning of the sampling rate will take some time as we will need to test it rigorously to start when needed (20+ mph only) and end when needed. This sample rate also needs to be variable to account for speeding up as the samples need to be 300mm apart.
Internal storage implementation	10hrs/wk, 1 wk	This part should not take too long as we will just be storing the values obtained in the previous task to the internal storage (SD card).
Sending to server	10hrs/wk, 2 wk	Sending information to the server will first require setting up post requests to the back-end server. Then we can just filter in these post requests one after another.
Storing in database	10hrs/wk, 1wk	Storing values to the database will only require setting one up and routing in the information. New values should overwrite old values.
IRI calculation implementation	15hrs/wk, 5wks	This will take the most time as it is more unfamiliar to us. It will also require a great deal of testing to make sure it is accurate.
Front-end web application	10hrs/wk, 4 wks	The front-end web application will be very simple and just provide a map with IRI and speed values along with some color coding. This is again a stretched goal and we may have to continue other work instead if it gets bumped back.

4.4 Other Resource Requirements

One potentially overlooked resource is the manpower required to assemble multiple devices and set them up for IRI readings and server communication, as well as vehicle installation. In addition to this, collecting data on roads throughout Iowa we will require the assistance of Iowa DoT employees.

4.5 Financial Requirements

The estimated cost for our device $100. The cost of using a cloud server to receive data and a mobile data plan to transmit data from our device is not included in this estimate. As mobile data infrastructure changes (2G being discontinued in favor of 4G) could also increase the cost of this. However, setting up a singular data plan for a range of devices could offset these costs.

5. Testing and Implementation

5.1 Interface Specifications

The black box will collect accelerometer data, timestamps, and GPS coordinates, and send this data over the 2G wireless network via http to a cloud server. The raw data will be stored on an SD card which is filled to capacity as data is collected. As new data is stored on the SD card, it is buffered and sent to the server. Data is not deleted from the SD card and remains as a backup in case the wireless transmission fails. The data will be transferable via USB to the server for processing.

The accelerometer and GPS/GSM module can use 3.3V or 5V power input which will be provided by the Arduino, which is powered by the car. The accelerometer and GPS/GSM both use digital I/O, as does the microSD module, and will transmit data to and from the Arduino using the SDA pin.

The Arduino Uno recommends between 7V and 12V DC power input via USB-B. This is achievable using the 12V cigarette lighter found in most cars and will be the preferred method of power for testing. Since it's possible that a vehicle does not have a 12V cigarette lighter, it's also possible to power the black box using an OBD port which is used for diagnostics on all modern cars. OBD ports output 12V DC, so either method of powering the Uno will be sufficient.

5.2 Hardware and software

The hardware used is an Arduino UNO with accelerometer and miscellaneous modules. For the purpose of testing, GSM communication/testing will take place after the sensor setup and after we verify the accuracy of IRI calculations.

Arduino Software IDE will be used to analyze the data taken and determine the accuracy of sensors and IRI calculations.

Sample tests for Arduino components and sensor data can be done in a lab setup. To test our IRI calculations, we can compare sample data taken from our device with verified IRI data taken from the truck provided from the Iowa Department of Transportation. The truck uses a more sophisticated method of IRI calculation, so we will have to determine an acceptable range of error to compensate for this, after we are confident in our calculations.

5.3 Functional Testing

System Tests

- GPS: data accuracy by checking it with other devices
- GPRS: by pinging server with it
- Accelerometer: compare with data recorded by cell phone, compare freefall readings versus calculated expectations
- SD card: check the data stored by creating a test file, test file format created by other components stored on SD card match up with expected data

Unit Tests

- Test data collection on a road for which the IRI is known or measured by accurate expensive systems
- Make sure all collected data reaches the server
- Run variety of dummy data through back-end
- Run variety of dummy data through front-end

Integration Tests

- Back-end will be tested using dummy data then real data with comparisons
- Front-end will be tested using dummy data then real data with comparisons
- Front-end tested using webdriver

5.4 Non-Functional Testing

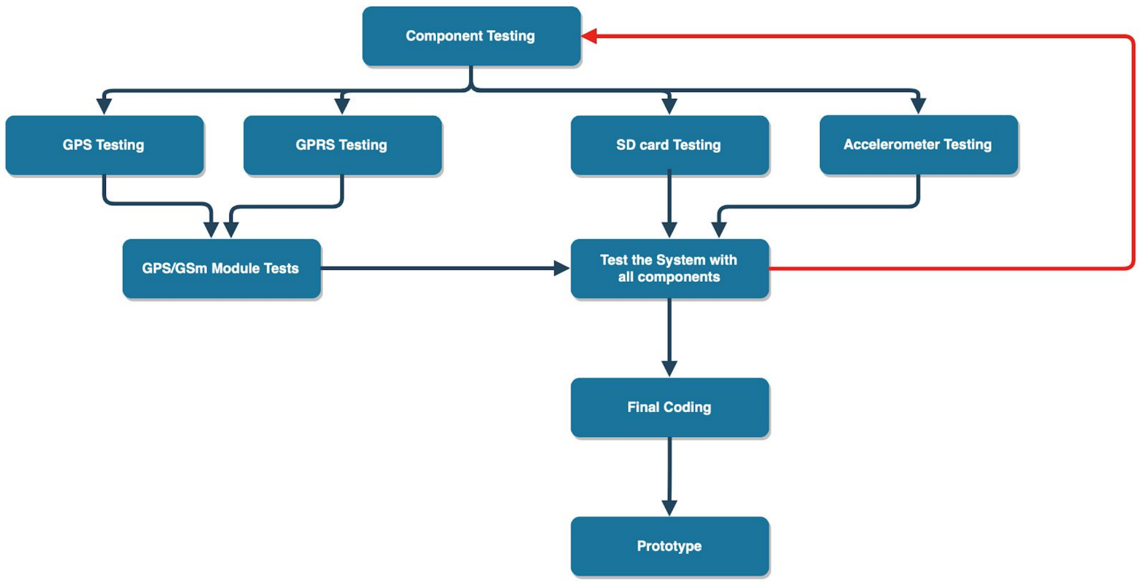
Performance: Performance testing will be conducted as a way of verifying the accuracy of our devices sensors and IRI calculations.

Security: The IRI data will be stored and displayed on a web server, some security is necessary, so only verified users will be able to access this data.

Usability: The usability of our devices will be seen in the setup of our device for use in a new vehicle. After mounting the 'black box' to a vehicle and setting up power, the device should take, and communicate data without user input indefinitely.

Compatibility: The IRI calculations we are using assume the 'Golden car model.' This assumes an ideal environment for car suspension and dampening, as it is unrealistic to obtain the exact data for each unique car. Instead, the quarter-car model with ideal parameters is used to obtain close to accurate results.

5.5 Process



5.6 Results

This section will contain all results including failures and successes. It will have tables of raw data processed into IRI data for our various iterations of code, as well as examples of raw data collected by the black box at various stages. Figures of the front-end will be included here, demonstrating portions of road and their IRI values.

6. Closing Material

6.1 Conclusion

The goal for this project is to create a black box system to calculate the International Roughness Index for the Iowa Department of Transportation. This black box system will be implemented into Iowa DOT's fleet of vehicles to provide the department data which they can use to prioritize road work in the future.

So far we have researched extensively on parts and the IRI calculation. We have gotten the parts already to mess around with them before the end of the semester. We have also started using Microsoft Azure which we will be using as the cloud server for the project. Once the beginning of next semester rolls around, we will start working hard on the implementation of our project. This will include wiring up the arduino and developing the code for it as well as the code for the server. We will then likely make a small website that uses the server to display relevant information.

6.2 References

Abudinen, Daniel & Fuentes, Luis & Carvajal-Muñoz, Juan. (2016). Development of Thresholds for Travel Quality Assessment In Colombian Urban Roads.

Alvarez, E. J. Z. (2016). A Discrete Roughness Index for Longitudinal Road Profiles. *Virginia Polytechnic Institute and State University*. Retrieved from https://vtechworks.lib.vt.edu/bitstream/handle/10919/64452/Zamora_Alvarez_EJ_T_2016.pdf.

Roadroid. (2019). V2PRO.

Sparkfun.com. (2019). Accelerometer, Gyro and IMU Buying Guide - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/pages/accel_gyro_guide [Accessed 20 Nov. 2019].

Yuchuan Du, Chenglong Liu, Difei Wu, and Shengchuan Jiang, "Measurement of International Roughness Index by Using -Axis Accelerometers and GPS," Mathematical Problems in Engineering, vol. 2014, Article ID 928980, 10 pages, 2014. https://doi.org/10.1155/2014/928980.

6.3 Appendices

Appendix A Datasheets

PMOD NAV 9-AXIS IMU/BAROMETER
https://www.st.com/content/ccc/resource/technical/document/datasheet/1e/3f/2a/d6/25/eb/48/46/DM00103319.pdf/files/DM00103319.pdf/jcr:content/translations/en.DM00103319.pdf

ARDUINO UNO REV3 SMD
https://media.digikey.com/pdf/Data%20Sheets/Arduino%20PDFs/A000073_Web.pdf

Adafruit FONA 808 Cellular + GPS Shield for Arduino
https://cdn-learn.adafruit.com/downloads/pdf/adafruit-fona-808-cellular-plus-gps-shield-for-arduino.pdf

MicroSD card module for Arduino
https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0229_Web.pdf

Appendix B Diagrams, Graphs, Charts

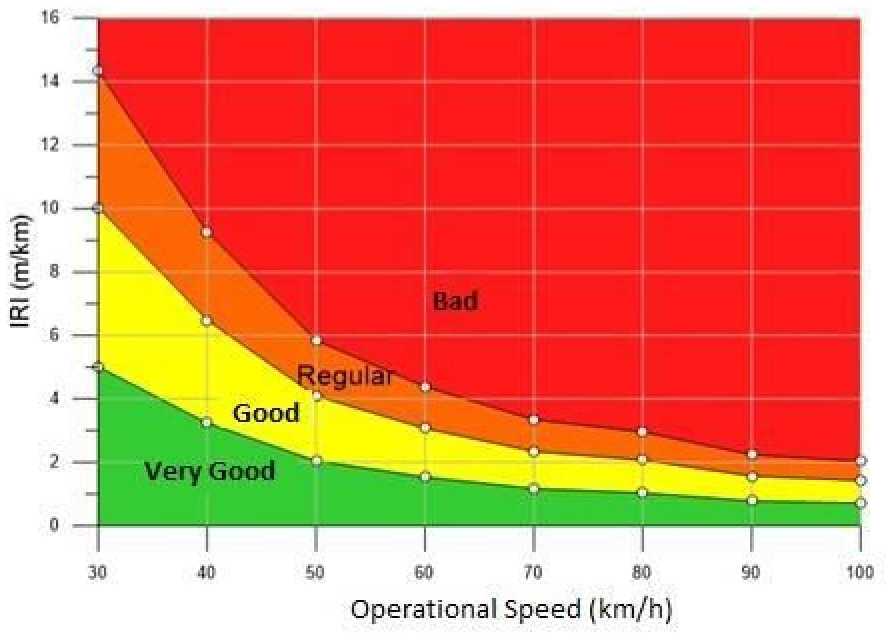


Figure 1: IRI vs. Operational Speed with Regions of Acceptability, from Abudinen et. al.