Spring 2016

Pothole Reporting System

Alissa L. McGill
University of Akron, alm161@zips.uakron.edu

Brian G. Simmons
University of Akron, bgs13@zips.uakron.edu

Sean D. Querry
University of Akron, sdq3@zips.uakron.edu

Elizabeth J. Hammell
University of Akron, ejh37@zips.uakron.edu

Alissa McGill
Honor Project Report
4-18-2016

Pothole Reporting System

The pothole reporting system is a device that is meant to be attached to the bottom of a vehicle for the tracking and logging of potholes the vehicle passes over. The system uses ultrasonic sensors to detect abnormalities in the road, and then sends the GPS location of the detected pothole to the user's smartphone device. From there, the smartphone logs the gps locations to a remote database where they can be reviewed by individuals who are in charge of improving road conditions. The smartphone application also has the capability to display the locations of any previously logged potholes on a map, so that pothole filled roads may be avoided by the user.

My contribution to this project included the development of the android application, the creation and management of the remote database, and parts of the bluetooth connection from the android application to the system's microcontroller. The development of the android application included designing the user interface that would be seen on the smartphone, the creation of a bluetooth socket to communicate with the microcontroller, and the google maps interface that would display the GPS locations of the potholes to the user. The android code also was in charge of interpolating data to factor in the speed of the car and the delay in the bluetooth connection. The GPS locations of the potholes and speed of the vehicle were acquired using the Google Maps API for android devices. The android application also included some code that would allow the potholes to be logged to the database using an http request. The creation of the database required setting up a server using Google's cloud console. A small database to hold the pothole location data was created using MySql. I wrote php code so that the android application could call these php pages on the server to submit changes to the database using Sql commands. The bluetooth connection was managed by the Larid BT900 bluetooth module, which was equipped with its own microcontroller to allow for easier configuration of the bluetooth connection. I wrote smartBASIC code for the bluetooth module, which allowed for the passing of information between the android application to the microcontroller.

Pothole Reporting System

Final Design Report

Design Team J

Elizabeth Hammell, EE, Hardware Manager
Alissa McGill, CpE, Software Manager
Sean Querry, EE, Project Leader
Brian Simmons, CpE, Archivist

Faculty Advisor: Seungdeog Choi

Wednesday December 2, 2015

Table of Contents

List of Figures

List of Tables

List of Sample Code

List of Equations

1. Abstract

The purpose of this project is to create a pothole detection device that can be attached to the underside of a commercial vehicle. Potholes cost motorists around 6.4 billion dollars annually, thus demonstrating the need for a system to aid with the detections and reporting of potholes. The four systems we needed to consider for the implementation of this project were the power system, the sensing system, the data processing system, and the reporting and logging system. Power pulled from the vehicle will enable the sensors and data processing module. The data processing module will analyze the readings from the sensors and output pothole data to the logging and reporting system. The logging and reporting system, located on an android mobile device, will store the pothole locations on a cloud server.

[BS, AM]

2. Problem Statement

2.1. Need

Potholes are dangerous, damaging obstacles on the road for drivers. Not only are they jolting and uncomfortable, but they are a large cause of vehicular accidents every year. There is a need for a way to accurately sense these potholes while driving and report them to the appropriate authorities.

[BS, AM]

2.2. Objective

Our proposed device is a universal system that can be installed on any passenger vehicle. It would detect the presence of potholes and report those potholes to the driver's smartphone. An app on that phone would then upload the pothole locations to a remote database that stores pothole location data for all users of this app.

[BS]

2.3. Background

Patent Search

Three patents were found that are related to this topic. The first of which is US 20140196529 A1, published on July 17, 2014. [1] This patent refers to a system where multiple devices gather information about the location of potholes and send the data to maintenance departments that would then dispatch repair crews. This patent goes a little beyond the scope of our proposed project, because it includes multiple forms of data acquisition whereas the proposed project would be a single device attached to a vehicle.

The second patent, US 5294210 A, was published March 15, 1994.[2] It refers to an automated pothole repair system that would detect a pothole, alert / slow / halt the vehicle, and patch the pothole. This is relative to our project in terms of pothole detection, but not so much in terms of the repair functionality.

The third patent, US 20140355839 A1, was published on December 4, 2014. [3] It refers to a mobile apparatus that would analyze images taken of a surface for degradation and, as a result, potholes. This separates from the proposed project where it keeps a record of the geolocation correlated to the image, and would then be able to generate trends from the data it gathers.

[BS]

Article Search

The first article we found was about using laser imaging to detect the location and the severity of potholes.[4] The system in the uses the laser to light up the road, which would help eliminate false positives caused by shadows and dark marks on the road. Laser imaging seems like a promising direction for us to take in our project.

The second article we found uses a laser line stripper that sends out a plane of light that intersects with objects which is in turn viewed by a camera. [5] This further strengthens the idea of using laser imaging for our own sensor. This article also describes using GPS to track the location of the pothole, and how these collected locations can be sent to a server via a Radio Data System. The system in this project also broadcasts a warning message to other nearby vehicles, warning them that there is a pothole nearby. This might be an interesting path to explore for our project requirements.

The third article describes a different way of detecting potholes then the first two approaches. The third approach uses a physical optics model to determine the backscatter response of road surface faults. [6] This way would also help to eliminate false positives due to dark marks and shadows on the road, because it uses radar instead of imaging. The sensor that would pick up the backscatter response would need to be very accurate to detect potholes that are shallow, because the change in distance for such potholes might be hard to detect.

The fourth article we found uses accelerometers to detect potholes that the car hits, as opposed to scanning the actual road beneath the car in any way.[7] This might be beneficial to combine with one of the above methods as a way to help determine between false positives and true potholes.

[AM]

3. **Marketing Requirements**
    1. The system is compatible with most vehicles.
    2. The system detects potholes in real-time as the host vehicle drives over them.
    3. The system detects potholes large enough to potentially cause damage if hit.
    4. The system is able to distinguish between potholes and other road surface anomalies.
    5. The system accurately records and logs time stamped GPS coordinates of the potholes it encounters.
    6. The system is autonomous, requiring no user input to function.

[EH, SQ]

4. **Objective Tree**

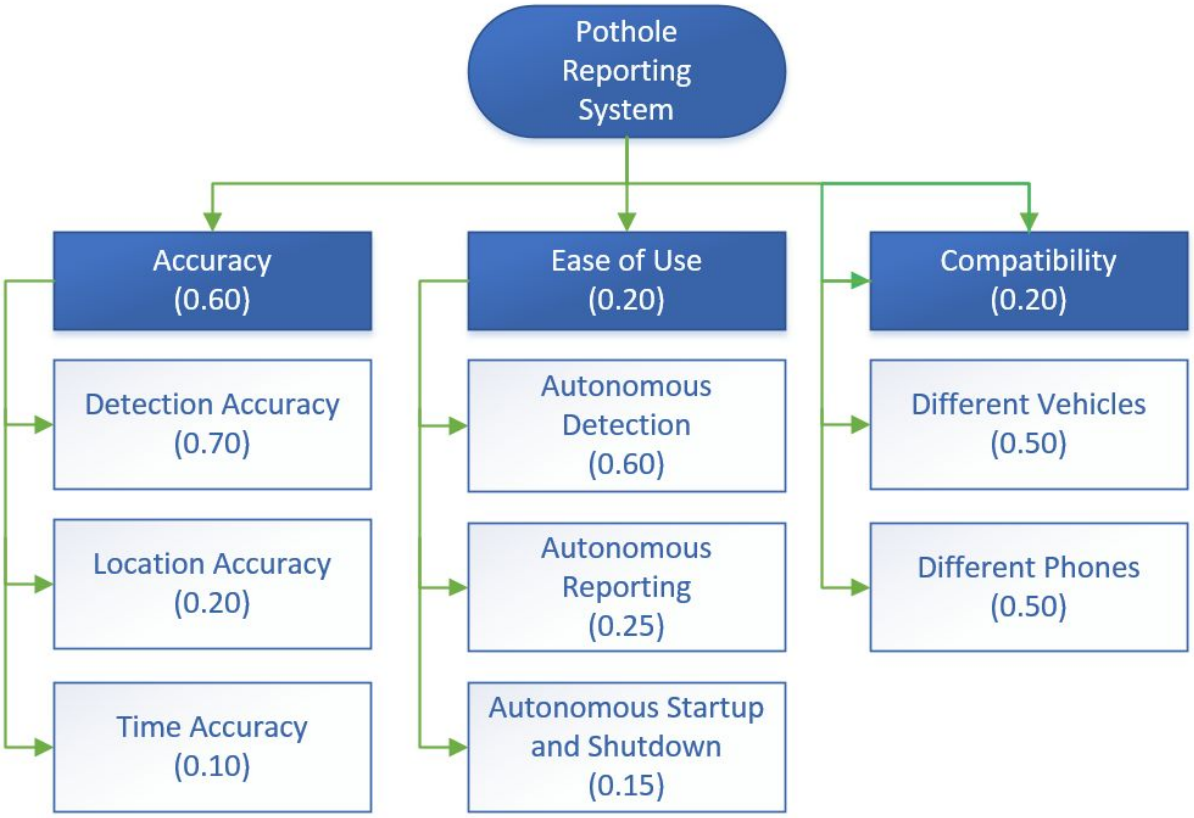

Figure 1: Objective Tree

[EH, SQ]

5. Design Specifications

5.1. Engineering Requirements

Table 1: Engineering Design Requirements

Engineering Requirement	Marketing Requirements	Justification
System can detect potholes at least at 8" in diameter	2,3	Potholes of this size are sufficiently large to potentially damage the vehicle.
System can detect potholes at speeds up to 70 mph	2,3	System should be fast enough to detect potholes on the freeway.
Sensor array must have a detection area at least 60" wide	1,2,3	Most cars have a width around 70". The system should be able to detect potholes along the width of the car.
System must function over voltage range of 12V - 15V	1	Vehicle system voltage varies within this range during operation.
System must withstand transients as strong as +/- 40V	1	Strong transients are not uncommon during vehicle operation, and could damage sensitive electronics.
Power consumption <0.1W when engine off	1	System should not interfere with vehicle operation; that includes draining the battery when not in use.
Recorded pothole locations must be accurate to within 30 ft	5	Two car-lengths is sufficient accuracy for municipalities to locate and repair potholes.
Recorded pothole detection time must have maximum resolution of 1 minute	5	This is sufficient resolution for municipalities to determine precise reporting times.
System must wirelessly connect to a cell phone up to 15 ft away	5	Distance between system and cell phone could vary based on system placement.
System must start and stop operation automatically	1,6	Minimized required user effort promotes system acceptance and deployment.
System must detect and report potholes automatically	6	Minimized required user effort promotes system acceptance and deployment.

[EH, SQ]

6. Accepted Technical Design

6.1. Theory of Operation

The sensors and module will be mounted onto a motor vehicle, pulling power from the vehicle itself. As the vehicle moves over a pothole, the sensors will continue to scan the road and send the data to a microcontroller within the module whenever a drastic change in road condition is present. Via Bluetooth connection, a signal indicating the detection of a pothole along with a timestamp will be sent to an Android Application on the passenger's mobile device. This information will then be uploaded to a cloud server, which will map out all of the potholes it receives. This system is susceptible to a number of different false-positive readings, including (but not limited to) snow and driveways / parking lots that are not the responsibility of the local government. It is for this reason that a temperature sensor will be implemented, and the system will not sense for potholes under freezing temperatures. Also, speed data will be sent from the mobile device via the Bluetooth connection to the module so that the system will only function above speeds of 20 mph.

[BS]

6.2. Hardware

General Overview

The pothole reporting system hardware is divided into four categories: power, sensing, data processing, and reporting.

The power subsystem consists of voltage regulators, filters, and a voltage supervisor. Its main purpose is to serve as an interface between the automotive battery circuit and the pothole reporting system circuits; it will generate the power supplies for both the sensing and data processing circuits.

The sensing subsystem consists of sensors, transmission lines, mounting hardware, and possibly interfacing circuitry, depending on the type of sensor chosen. The purpose of this subsystem is to scan the road surface and detect potholes. Distance measurements will be converted into analog or digital signals and sent to the data processing subsystem.

The data processing subsystem consists of a microcontroller or microprocessor, a Bluetooth radio, a real-time clock, a custom PCB, and mounting hardware. This subsystem is responsible for interpreting the sensing subsystem data, determining whether a pothole is present, and notifying the reporting subsystem if a pothole has been detected.

The reporting subsystem consists of a smartphone containing a Bluetooth radio, a GPS radio, and a cellular radio. This subsystem takes data from the data processing subsystem via Bluetooth and matches that data with its corresponding GPS data. It then sends the complete pothole location data to a remote server via cellular transmission.

[SQ]

Power

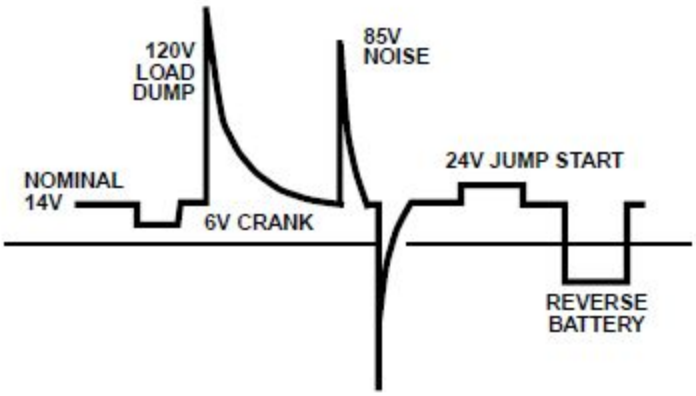

Figure 2: Test Pulses for Automotive Electrical Systems

Powering sensitive electronics in an automotive environment will require careful planning. In order to maintain compatibility with a multitude of vehicles, the best solution for tapping into vehicle power is to tie in directly to the vehicle's battery, as an under-the-hood battery is nearly universal in passenger vehicles. The issue with this is that the system will be exposed to an extremely dirty power circuit. A typical vehicle's electrical system is powered by an alternator, which converts mechanical rotational energy into an AC voltage, which is rectified and sent out to the rest of the system; even under ideal conditions, the alternator will produce ripple that must be considered.

Perhaps even more of an issue than the alternator is the number of various loads that are present in the electrical system. From relays, fuel injectors, and spark plugs, to wiper motors, blower motors, audio systems, and lighting systems, the switching of loads is constantly introducing transients into the system. SAE research has shown that transients from 25V to 125V are not uncommon. [8] Figure 2 shows some typical automotive test pulses as defined by standards such as ISO 7637-2 §A.5; these pulses are designed to replicate worst-case scenarios encountered in real 12V automotive systems. [9] Because this project's power subsystem will be connected directly to the host automobile's battery, the subsystem must be designed to withstand such pulses.

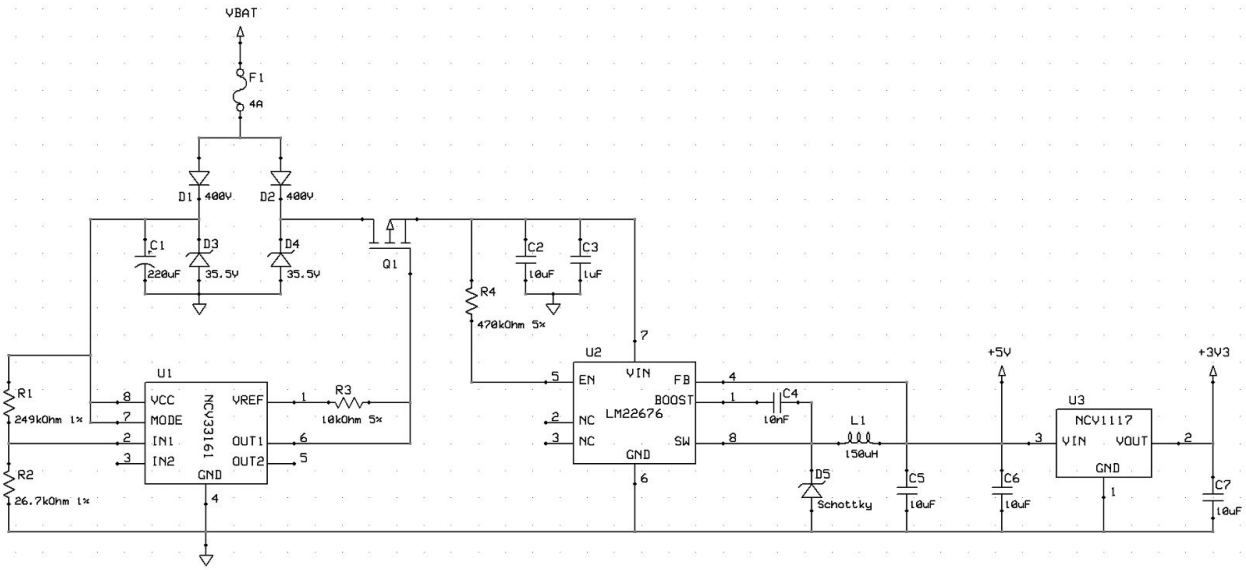

Figure 3: Power Subsystem Circuit Diagram

Figure 3 shows the circuit diagram for the power circuitry used for this project. Before designing this subsystem, anticipated maximum current draw was calculated based on the following data:

Table 2: Project Current Consumption Projections

Component	Current Draw	# of Components	Total component current draw	Data Source
Laird BT900	85mA	1	85mA	Datasheet
SainSmart HY-SRF05	7mA	18	126mA	Observed
Microchip PIC24EP64GP206	42mA	1	42mA	Datasheet
Other	100mA	N/A	100mA	Hypothesized
TOTAL			**353mA**	

Note that an estimated 100mA draw was added to account for quiescent currents and passive component current draw, as well as to account for current draw fluctuations due to temperature variation. However, because this subsystem will require a custom PCB, significant overhead was factored in to the design in order to keep it flexible; should any sensing or processing subsystem hardware need to change, the power subsystem should still be able to provide adequate power. Thus a current consumption target of 3A was chosen for this subsystem.

Thus, a 4A fuse is placed between the host vehicle's battery and the input to the subsystem based on the following equation, based on a common rule-of-thumb:

$$I_{rated} = I_{max} * 1.35 \approx 4A.$$

Equation 1: Calculation of Fuse Value

After the fuse, two polarity protection diodes will be employed, as shown in Fig. X; D1 is a Diodes, Inc. SBR1U400P1, and D2 is a Vishay MURS340HE3. ISO 7367-2 indicates that automotive electronic systems must be able to withstand pulses with magnitude as high as +/- 150V. [9] Thus both diodes were chosen for their peak reverse voltage values of 400V; this value provides a considerable safety margin to ensure that the project's electronics aren't exposed to reverse polarity conditions. Note that two separate diodes are used so that the voltage supervisor IC has its own input branch. This architecture was chosen because a diode's forward voltage drop is directly related to the current flowing through the diode, as shown in Figure 4.

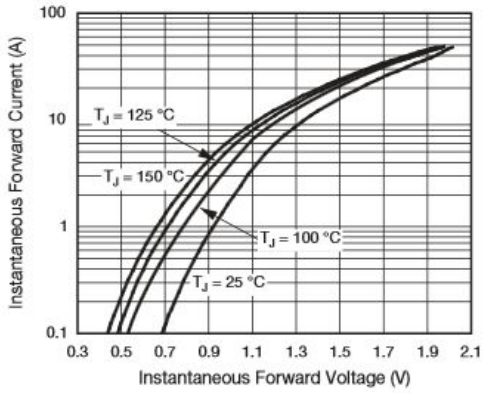

Figure 4: Forward Voltage Characteristics of Vishay MURS340HE3

Because the current consumption of the system can vary wildly based on whether the Bluetooth radio is transmitting, the second input branch was implemented to ensure that the voltage supervisor has a nearly-constant current, providing predictable behavior; the voltage supervisor is discussed in greater detail below.

To complement the polarity protection diodes, there are also two transient voltage suppression (TVS) diodes. While the polarity protection diodes protect the circuitry against reverse polarity conditions, the TVS diodes protect against overvoltage conditions. Transient spikes in excess of +100V could easily cause catastrophic damage if not controlled. Thus, for each input branch, a Vishay 3KASMC22AHM3 is employed. This diodes begin to enter the breakdown region at 24.4V, and they insure a maximum clamped voltage of 35.5V. With these TVS diodes employed, components rated for 40V inputs can be placed on the input-facing nodes, making a practical design realizable.

On the first input branch, an ON Semiconductor NCV33161 Voltage Monitor IC is employed. This IC monitors the voltage on its IN1 pin, comparing it against an internal

reference threshold of 500mV. The IC incorporates hysteresis on its input comparators, allowing for the design of separate rising and falling threshold levels, as shown in Figure 5.

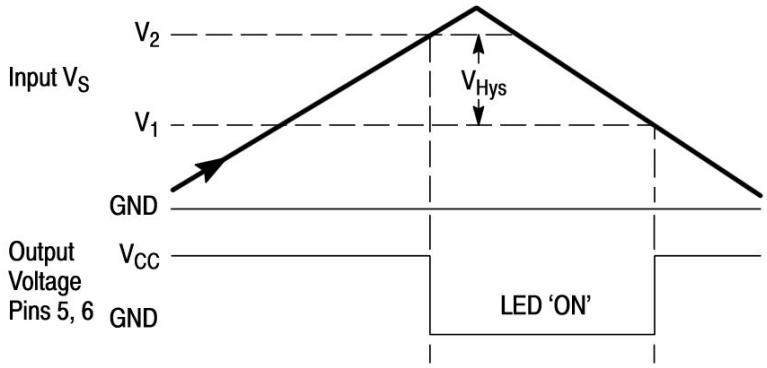

Figure 5: Input-Output Relation of NCV33161 in Inverting Mode

Thus, threshold levels V1 and V2 were chosen given the following equations:

$$\frac{R_2}{R_1} = \frac{V_1}{V_{th} - V_H} - 1 \qquad \frac{R_2}{R_1} = \frac{V_2}{V_{th}} - 1$$

Equation 2: Threshold Level Equations for NCV33161

Given nominal values Vth = 1.27V and VH = 25mV, and given a desired lower voltage threshold of V1 = 12.85V, it can be calculated that V2 = 13.1V. Given these thresholds, and given a desired total bias resistance between 250kΩ, the following equation was used to calculate the necessary bias resistors:

$$R_2 = 9.321R_1.$$

Equation 3: Theoretical Bias Resistor Ratio

Using this equation, and choosing from the standard values of resistors with 1% tolerance, the desired bias resistor values were found to be RB1 = 249kΩ and RB2 = 26.7kΩ. This configuration yields the ratio

$$R_2 = 9.326R_1,$$

Equation 4: Actual Bias Resistor Ratio

which gives a nominal bias resistor ratio error of 0.005.

Also at the input of the voltage supervisor, a 220uF bulk capacitor is employed to introduce delay into the switching of the supervisor. Using the equation

$$V = V_0 e^{-t/RC},$$

Equation 5: Capacitor Discharge Equation

and given V = 12.85V, V0 = 13.6V, R = 275.7kΩ, and given a desired discharge time of 10mS, C was calculated to be 176uF. Accounting for tolerances and temperature variation, the 220uF capacitor was chosen. This device will yield a 10ms delay in the switching behavior of the supervisor, preventing short negative transients from causing the IC to disable its output.

The output of the voltage supervisor is pulled to the NCV33161's pinned out 2.54V reference through a 10kΩ resistor when not active. This output is connected to the gate of a P-channel enhancement mode MOSFET. When the vehicle is started, and the battery voltage rises from 12.6V to approximately 14V, the IC pulls its output low, activating the MOSFET, which connects VBAT to the input of the switching voltage regulator, thus allowing the system to start up.

The switching voltage regulator used is the TI LM22676. This component steps down the input voltage to a constant 5V DC output. It was chosen for its high maximum input voltage, its low accompanying part count, and its ripple and efficiency characteristics. The regulator's output voltage ripple is given by the equation

$$V_{ro} \approx \frac{(V_{in}-V_{out})\cdot V_{out}}{8\cdot V_{in}} \cdot \frac{1}{F_{sw}^2 \cdot L \cdot C_{out}}.$$

Equation 6: Output Voltage Ripple Formula for LM22676

Given the parts shown in Figure 3, L = 150 uH, Cout = 10uF, Vin = 14V, Vout = 5V, and Fsw = 500kHz. This gives an output ripple voltage of approximately 1mV, or 0.2%. While these parameters will change due to tolerances, temperature variations, and effects of the surrounding circuitry, this calculation is still a valid indicator of the regulator's performance. Due to the low ripple of this unit, a linear post-regulator is not needed for the +5V rail, simplifying the cost and complexity of the power subsystem. Furthermore, using TI's WEBENCH design software, this design was found to be 78% efficient at the project's nominal projected current draw of 0.3A. By comparison, a linear regulator in this scenario would be approximately 36% efficient.

While the +5V rail powers the sensing subsystem for this project, a separate +3.3V rail is needed to power the data processing subsystem hardware. Furthermore, since an RF communication module is employed in the data processing subsystem, ripple rejection is critical. To generate this rail, an ON Semiconductor NCV1117 is used. This low-dropout, linear regulator steps down the +5V output from the LM22676 and generates a fixed +3.3V output; it is capable of supplying up to 1A, or roughly 8x the projected need. This part was chosen for its nominal 0.003% output noise, its high 64dB ripple rejection, and its line and load regulation characteristics.

All active components chosen for this subsystem AEC-Q101 qualified for automotive use; they have extended junction temperature operating ranges of -40 to +120 degC. All components are readily available, and the total BOM cost for this subsystem is just over $10, excluding the PCB. The design of this subsystem makes it simple, reliable, flexible, and inexpensive, providing a solid foundation for the other subsystems.

[SQ]

Sensor Options

For the system, a few sensors are required to identify temperature of the environment and the possibility of a pothole in the road surface.

External temperature will be monitored by a temperature probe and will enable the microcontroller if the system is within its operating range. To minimize cost and add to ease of manufacture the probe will be chosen to have the same operating voltage as the distance sensor. If any additional discrete or integrated circuits are necessary they will be added to the interfacing circuitry board shared by the temperature and distance sensors.

The distance between the car mounted sensors and ground will be measured by an array of sensors. The number of sensors needed is directly dependant on the type and model of sensors selected. In order to fully scan the area between the wheels of the vehicle the sensors will be placed so that the field of view surveys enough road surface to detect an 8 inch pothole. The presence of a pothole will be denoted by a sharp change in distance reported by the sensors. Thus sensors must be selected that have a range of 4 to 12 inches to be practical on a variety of vehicles. Additionally the sensors need to be operable with a wide variety of road surfaces, be cost effective, and provide an output that is easy to process.

The accuracy of the system depends strongly on the type of distance sensors selected, thus several options were explored. These options include infrared (IR) triangulation, ultrasonic, time of flight IR, and 3D imaging sensors. LIDAR is not being considered simply due to cost. The below table was instrumental in making a decision that best suits the needs of the project. Table 2 weighs the requirements of the project against each other in order to compare each type of sensor being considered. For each requirement the sensors are compared to each other, with larger numbers representing higher performance compared to the other options. The ultrasonic ended up with the highest value and thus was chosen for the system. This type of sensor is good all-around: it has good range, speed and requires little in the way of processing the output.

Table 3: Initial Decision Matrix

Requirements	Weight	Type of Sensor				Total
		IR Triangulation	Ultrasonic	Time of Flight IR	3D Imaging	
Range	0.25	0.3	0.3	0.1	0.3	1
Speed	0.35	0.05	0.28	0.32	0.35	1
Field of View	0.05	0.15	0.25	0.15	0.45	1
Mech. Implementation	0.05	0.3	0.25	0.3	0.15	1
Processing Required	0.15	0.23	0.36	0.36	0.05	1
Cost	0.15	0.32	0.32	0.32	0.04	1
Total	1	0.1975	0.3	0.2615	0.241	

[EH]

Ultrasonic Distance Sensor

Ultrasonic sensors are commonly used in cars as back up assistants to notify the driver of an object behind the vehicle that would otherwise be in the driver's blind spot. In this type of application the sensors are only activated when the car is shifted into reverse. Therefore the car is moving slowly and the sensors are not being exposed to large amounts of vibration. For ultrasonic sensors to work for a pothole sensing system, they would need to be isolated from vehicle vibration during routine driving, including down rough roads and at highway speeds. Other than its susceptibility to vibration, the ultrasonic sensor is nearly ideal for the pothole sensing application. There are many similar sensors; the one chosen for this project was the HY-SRF05 because of its competitive pricing and fast shipping options. This sensor has a range from 0.8 in to 3.3 yd and a speed that varies with the distance to an object. The maximum speed is the timeout value of 30 ms. The speed we are aiming for is 12 ms based on seeing an 8 in pothole while traveling at 70 mph.

This sensor includes additional circuitry that converts the echo information into a distance that is digitally output. With this type an array of sensors would be set up on the underside of the vehicle and send data to a project box where the data can be processed. See Figure 2 for a mockup of the car mounted array. The sensors are relatively inexpensive and have a wider field of view than the IR sensors allowing for fewer sensors in the array under the car. Additionally, the ultrasonic sensors work well with rough surfaces due to the nature of how the sensors work.

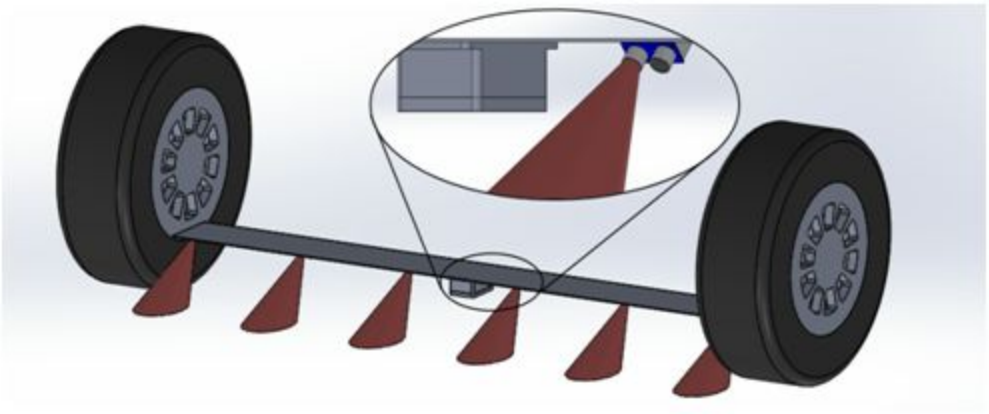

Figure 6: Preliminary Hardware Mounting

To allow for a more accurate system the sensors will be mounted at an angle. If the sensors were directed straight down at the pavement the return pulse could echo back and forth between the car and the ground several times before dissipating. Additionally if a small pothole is present in the field of view of the sensor, the echo coming off the pothole might not register at the receiver because the echoes off the road surface will get there first. For these reasons the sensors shall be mounted at an angle. Figure 3 below shows several mounting angles and how the sensor output (grey) is reflected off of the road surface and how the echo (pink) propagates.

Figure 7: Sensor Mounting Angle Comparison

The selected angle needs to be large enough to prevent the echo from being reflected back to the sensor. These ultrasonic sensors advertise that the field of view is 15º. The initial mounting angle for the sensors was chosen to be 45º because it is easy to implement and because the beam width is sufficient for the needs of the system. Figure 4 shows the values associated with the sensors being mounted at this chosen angle.

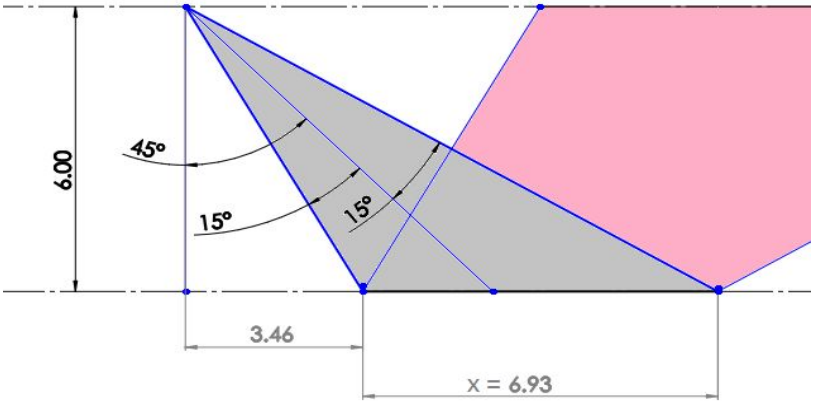

Figure 8: Sensor Mounting Angle Values

If the sensors are mounted at a 45º angle and are 6 inches above the road surface calculations can be performed to find the width of the beam when it strikes the pavement which is given by the variable x.

$h = 6.00\ in$

$\theta = 45°$

$\theta_{max} = \theta + 15° = 60°$

$\theta_{min} = \theta - 15° = 30°$

$\tan\theta_{min} - \tan\theta_{min} = x/h$

$x = h\,[\tan\theta_{max} - \tan\theta_{min}]$

$x = 6\,[\tan(45° + 15°) - \tan(45° - 15°)]$

$x = 6.9282\ in$

Figure 9: Calculated Beamwidth on Road Surface

These results agree with those in the diagram and if the angle is changed can be repeated for different angles as necessary.

A final consideration for the mounting of the sensors is how the signal will be reflected back to the source. Figure 5 shows how a signal propagating from the sensor can strike a pothole and be reflected back to the sensor. This methodology relies strongly on the pothole shape and will be experimentally tested to verify that the potholes are being detected.

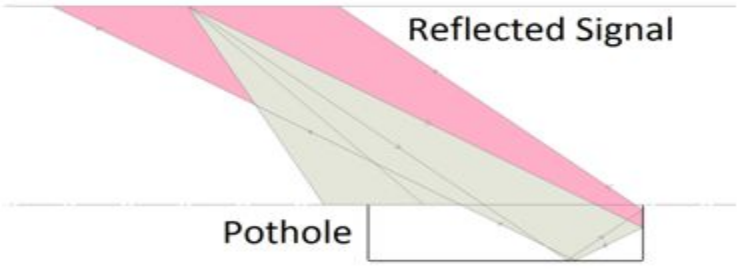

Figure 10: Reflected Signal Diagram

[EH]

Alternative Distance Sensors Considered

The IR Triangulation sensor method was another sensor considered. This type of system would be mechanically implemented in the same way as the ultrasonic sensors, but with the sensors being pointed straight down, not at an angle. How the IR triangulation works is that each sensor emits a beam of light which then travels to the surface that will reflect it and the reflected beam angle is measured and used to calculate the distance to the object. One of the drawbacks to this type is that the response depends on the color and texture of the material it is being reflected off of. For rough surfaces the sensors are not as reliable, and most road surfaces will be rough. These sensors were considered because they are cheap and come in a variety of styles that provide a range that would work with the system. Another drawback of this type is the speed of the sensors is low. For analog signals many models updated data every 20 or 40 ms, if the vehicle is moving at highway speeds this will not give an acceptable resolution. The digital sensors are faster, however, they take several data points and average them. The number of data points per sample is user defined, but the higher the number, the shorter the range of the sensor.

Another type of sensor, the Time of Flight (ToF) IR sensor, is an IR sensor that uses a clock to time how long it takes for a beam of light emitted to be reflected back. These sensors are more accurate than the triangulation IR sensors, but they also have a small field of view requiring a large array of sensors. The range is restricted, but research would need to be done to see if the range can be slightly extended to better work in the pothole sensing application. These signals provide digital outputs that are more convenient than analog data provided by some triangulation IR sensors. They are not heavily influenced by ambient light or the material of the reflected surface. All in all they are a good fallback if there are problems implementing the ultrasonic sensors.

The final method considered is a 3D image sensor. This technology was made popular by the Kinect and has been further improved in the new Kinect v2. The system is a ToF IR sensor but with additional capabilities and drawbacks. The field of view is much wider (several feet), but the dead-zone is also much larger and would require the system to be mounded to the front or rear of the vehicle and be pointed out and down at an angle. Additionally, the Kinect v2 is an expensive sensor and would also require an additional adapter to be purchased. The accuracy and speed provided would be significantly better than all of the other options, but at a much higher cost. As a final note, the other sensors would all be compatible with similar mounting structures and the same microcontroller. This system may even require a CPU and needs its own specialty mounting system.

[EH]

Temperature Sensor

The sensor chosen for the system was Microchip Technology Inc.'s MCP9808 digital temperature sensor. This sensor has on operating range from -20°C to 100°C with an accuracy of ± 0.5°C. It requires an input voltage between 2.7 and 5.5V and has on

operating current of 200μA. This chip is enabled with I^2C communication and has an additional feature that will be useful for the pothole reporting system. One of the pins on the IC is an alert pin. This output can be configured to act as an interrupt if the temperature drifts below the selected minimum value. There are break out boards available for this particular device, however it is more likely that the necessary resistors, capacitors, and connector pins will be added to a specialized PCB design. Because the distance sensor is not set in stone the PCB design is also still being developed.

[EH]

6.3. Software / Firmware

Comparative Analysis of Sensors

As mentioned previously, there are four main concepts taken into consideration in regards to reading the surface of the road and sensing for potholes. These four methods are IR triangulation sensors, Ultrasonic sensors, IR ToF sensors, and 3D image sensors. Each method has their pros and cons, and each method requires modifications or conversions to the data gathered in order for it to be analyzed.

If IR triangulation sensors are implemented in this system, the main concept to take into consideration is analog to digital conversion, and the ability to filter out noise within sensor's readings. The signal being sent from the sensor to the microcontroller will be a voltage within a particular range, based on the specifications of the sensor. This voltage is an indication of the distance from the sensor to the object that its IR light is reflecting off of. If the voltage range is 5V to 10V, for example, and the sensor is sending the microcontroller a voltage of 7.5V, the object it is seeing is right in the middle of the distance range it is specified for (if the range was 10cm to 20cm, the object would be sitting 15cm away). This analog data, however, is of little use in the digital world. Therefore the use of an analog to digital converter is required to read in this analog voltage and output bits that correspond to the voltage, with an accuracy based on the number of bits in the converter and the range that it is required to cover. Often these converters are 8-bit, 10-bit, or 12-bit, where a larger number of bits allows for greater precision (commonly referred to as the resolution of the converter).

$$\text{Precision} = \text{Voltage Range} / (2^{\text{bit}}-1)$$

Equation 7: Converter Precision Equation

As shown in Figure 3, the higher the number of bits in the converter, the smaller the volts per "count" (count refers to one unit of digital precision from the converter, for example a 10-bit converter will have 1023 total counts). Once this analog voltage is converted to a digital value, it can be used for comparison. The whole system is based on reading in multiple values from sensors and comparing them to their direct neighbors in search of a jump or drop in counts. If sensors "A" and "B" are right next to each other, with "A" reading 100 counts and "B" reading 800 counts, in our application this would correlate to a drastic change in road surface distance, in many cases a pothole.

This method is not without its flaws, however. Noise in the signal due to the shaking of the vehicle, imperfect lighting conditions, and general road quality may lead to many false-positives and missed potholes. It is for this reason that the signal will need to be checked with various thresholds so as to make sure the reading makes sense. It will be a struggle to filter out all possible noise, which is one of the major downsides to using this method of detection.

If IR Time of Flight (ToF) sensors are implemented in this system, the output to the microcontroller will most likely be a digital signal, depending upon the sensor selection. In this case an analog to digital converter will not be necessary, and instead the important concept to focus on is the communication protocols used between the sensor and the microcontroller. The sensor selection has a large part in this decision, and as long as the microcontroller is compatible with the method chosen (I2C, SPI, etc.) then the implementation of communication will be fairly simple.

However, the data will still have noise associated with it and will need to be filtered to prevent as many false-positives as possible. In addition, a digital signal will have a delayed arrival at the microcontroller. This is due to the additional circuitry the sensor signal is processed through, in order to convert it to a digital signal. This method, while technically different from the IR triangulation method, is very similar to it when it comes to analyzing the data.

For the aforementioned methods the algorithms for determining if a pothole has been located will be fairly similar, once the data has been gathered and converted into a digital signal for comparison.

```
Sensor_Readings_Array [ ]
for (k = 0; k > number_of_sensors; k++)
{
    x = Sensor_Readings_Array [k];        //values will be checked
    y = Sensor_Readings_Array [k+1];       //through thresholds to
    if (abs(x - y) > pothole_threshold)    //make sure sensor hardware
    {                                      //is not malfunctioning
        pothole_flag = true;
        timestamp = Current_Time;
     }
}
```

Sample Code 1: Sensor Comparison Pseudo Code

In Sample Code 1, each of the sensor's readings are placed into an array. Once these readings are verified so as to prevent the analysis of faulty data (due to damaged, covered, disconnected, etc. sensors), each value will be compared to the value of the sensor directly next to it. If the difference between the two sensors is greater than a threshold that indicates the likelihood of a pothole, the time of detection is recorded and a flag indicating the detection of a pothole is raised.

The third approach, utilizing a 3D image sensor to detect potholes, is vastly different in many ways from any other method. Using a 3D image sensor would change how the potholes are detected, from taking pinpointed distance measurements to processing an image and searching for changes in depth of the road surface within that image. Because of this, a simple microcontroller will not be capable of analyzing this data, and a CPU will need to be implemented into the system in order to do so. Ideally, the image from a 3D image sensor would output to the CPU a depth map as a function of color. Utilizing color tracking software, the CPU would be able to see anomalies in the road due to their raised or lowered height correlating to a change in color on the depth map. This method, while promising, differs greatly from every other method in both theory and cost.

In Sample Code 2 shown below, pseudo code is written to give a look into how the imaging data may be processed in order to determine if a pothole is located within the image.

```
depth_map = GetImage();
Convert (depth_map, BGR, HSV);
Filter_Image (depth_map, Pothole_H, Pothole_S, Pothole_V);
Eliminate_Noise(depth_map);
Dilate_Whitespace(depth_map);
depth_map_ones = Image_Analyze_Whitespace(depth_map);
if (depth_map_ones > pothole_threshold)
{
    pothole_flag = true;
    timestamp = Current_Time;
}
```

Sample Code 2: Image Processing

Here is an overview of how the image processing pseudo code would function: The CPU will read in the color-based depth map from the 3D image sensor. This image then needs to be converted from a BGR (Blue, Green, Red) image to an HSV (Hue, Saturation, Value) image. Once converted, the image will be filtered through hue, saturation, and value measurements that are characteristic of potholes in a road surface. If a pothole is present, the image will now be black and white, with the white space representing a pothole. This image will now go through filtering and dilation, so as to eliminate noise due to general road conditions and emphasize the pothole itself. Once filtered, the image will be analyzed for whitespace, and if the whitespace seen is greater than the amount necessary for a pothole to be recognized a flag will be raised and the time will be recorded.

[BS]

Accepted Methodology

The final option is to utilize ultrasonic technology. At this point in time, this is the method that was chosen. It is similar to using an IR sensor in that the sensor is

measuring and sending back the distance to the ground. However, unlike an IR sensor an ultrasonic sensor is not sensitive to lighting conditions, which is a great benefit to this method. The ultrasonic sensor chosen will, when its trigger pin is raised high for at least 10 microseconds, send out a 40 kHz pulse and listen for a response. Here are a few oscilloscope images of what this signal with the response looks like.

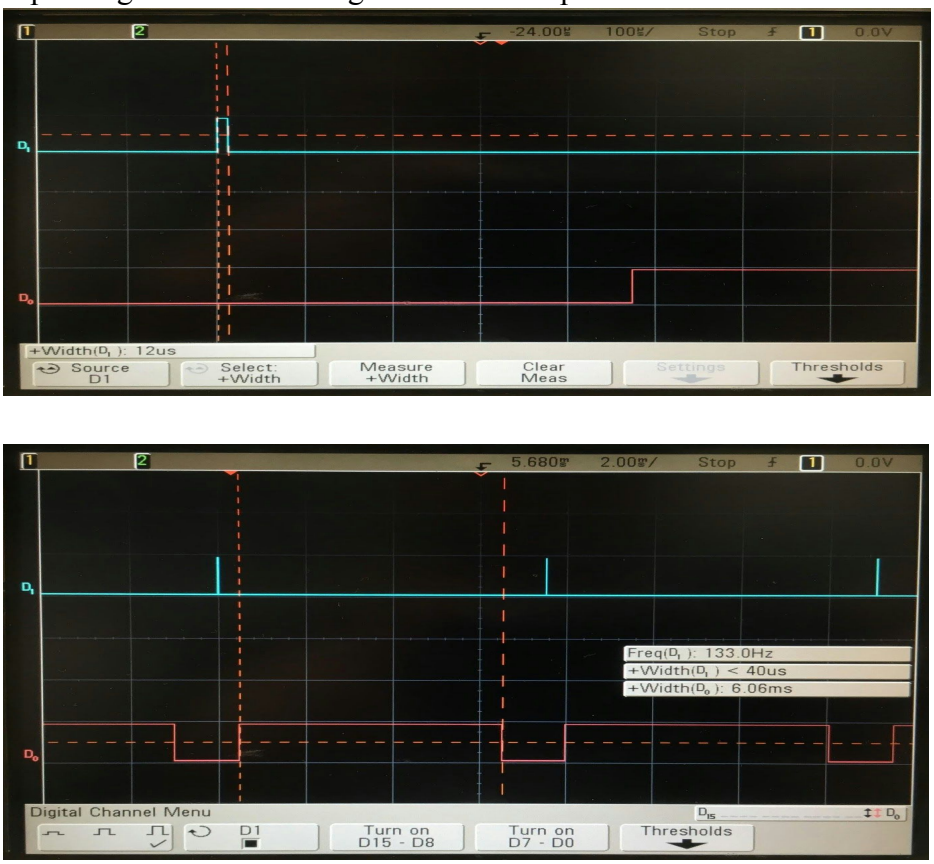

Figure 11: Oscilloscope Shots

The entire time the sensor is waiting for a response after sending the trigger (Blue), it raises it's echo pin (Red) high. Once the sensor receives a response, the pin is brought back down to zero. This time measurement can be equated to a distance measurement, which can be seen in Equation 8 below. If no response is heard, the sensor will time out at 30 milliseconds and wait for another trigger.

$$Distance\ in\ Inches = Time / 148$$

$$Distance\ in\ Centimeters = Time/58$$

Equation 8: Distance Conversion Equation

The good thing about this method is that it is very simple. One output pin is held high for 10 microseconds, and then the micro watches how long the input pin is held high by the sensor. The downside, however, is how much time is required for the sensor to time out - and here's why: As described previously, the sensors will be mounted at an

angle of 30 degrees. with normal road conditions without any potholes, the sensor will send out it's 40 kHz pulse and it will bounce off the road and away from the sensor. This means that under normal road conditions, the sensor will time out every single time it sends out a pulse to search for potholes, and will only get a reading back when the signal bounces off of the edge of a pothole. So under normal road conditions, the sensor will take just over 30 milliseconds to respond before it can be sent another command. The graph in Figure 12 shows the response time required to measure the distance to the ground at different speeds. It can be seen that as the speed increases, the response time needs to decrease.

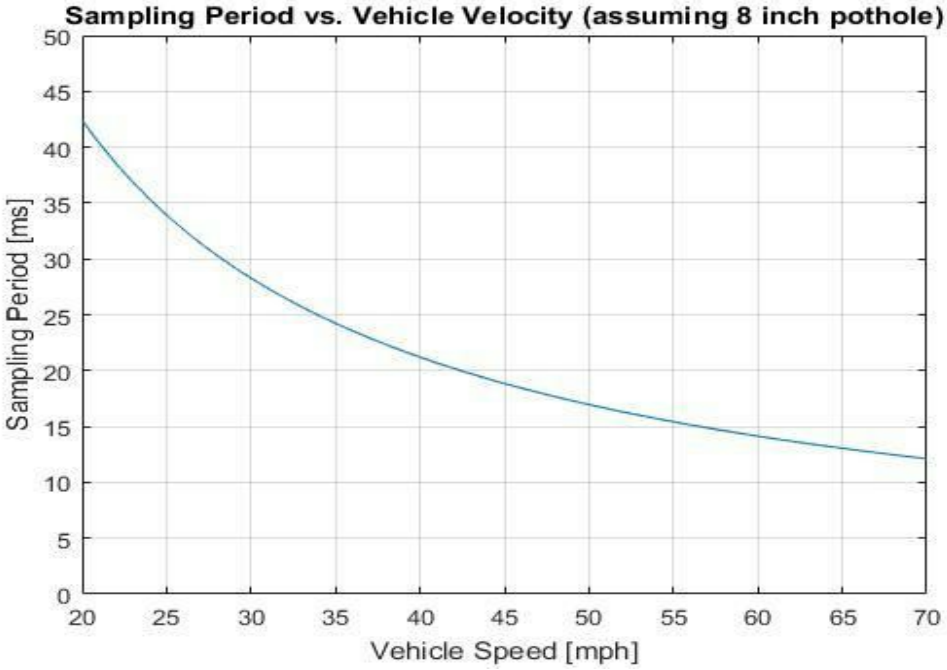

Figure 12: Speed Graph

There is a solution to this problem, however. Instead of having a single line of sensors, three separate arrays of sensors will be implemented, each one to be triggered at an offset of 10 milliseconds. This way, the rate at which samples will be taken is tripled, allowing for better precision at higher speeds.

[BS]

MCU/CPU Requirements and Considerations

For three of the four methods taken into consideration, a microcontroller would have the processing capabilities necessary to analyze the data from all the sensors. For the IR Triangulation method, the microcontroller needs to have enough analog to digital channels to read in data from multiple IR sensors, and be able to swap channels and resample fast enough so that the distance covered is negligible. For the IR ToF method and Ultrasonic method, as long as the microcontroller is compatible with the

communication protocols of the sensor, it should be able to analyze the digital signal without issue. In both cases, the microcontroller needs to be able to do the calculations in determining if a pothole is present all within the time it takes to get another reading from the sensors, and the microcontroller must also be capable of communicating with the wireless communication module being implemented.

For the 3D image sensor method of detecting potholes, a more complex and powerful system is required. It must be able to process a full image and determine if there is a pothole within the image. In this case, most likely a system similar to an Arduino or Raspberry Pi will be implemented, or if something stronger was required a windows machine would need to be used.

Since the method chosen is Ultrasonic sensors, a PIC microcontroller (specifically the PIC24EP64GP206) is being considered. It contains 53 separate I/O pins, which is very important due to the estimated 18 sensors required, each with two pins to be connected to the microcontroller. This specific PIC also has I2C and SPI digital communication capabilities to talk to the temperature sensor and Bluetooth module, as well as up to 7 timers (5 x 16-bit and 2 x 32-bit).

[BS]

Real Time Clock Selection

The real time clock (RTC) is a very important component within this system. It must be accurate and fast, because the time that it produces will be utilized by algorithms within the android application to determine as closely as possible the location of any detected potholes. For these reasons, the Microchip MCP79510 is the RTC that has been selected for this system. It communicates at 10 MHz with a clock accuracy of .01 seconds. It communicates via SPI, which is compatible with the microcontroller that has been chosen.

[BS]

Firmware Functionality

The Microcontroller will, for the most part, utilize a super loop to run its code after initialization. Various functions will be called to send and receive data from the sensors and mobile device. These functions that shall be described will either run once as a part of the initialization process or run frequently as the code passes through the super loop.

On the following page, a flow chart can be seen that accurately describes the flow of the firmware loop during the acquisition and analysis of data from three separate arrays of sensors. It can be seen that each array is only analyzed for 10 milliseconds before moving onto the next. Since no sensor should ever return a response time of less than 10 milliseconds unless a pothole is present, it is assumed that any value past 10 milliseconds will not be a pothole and can therefore be ignored. Any timestamps that are acquired are then sent over Bluetooth to a mobile device for logging and reporting. If an

interrupt has been received informing the system that it is either inside or outside of its operating range, this will decide if the system is enabled or disabled.

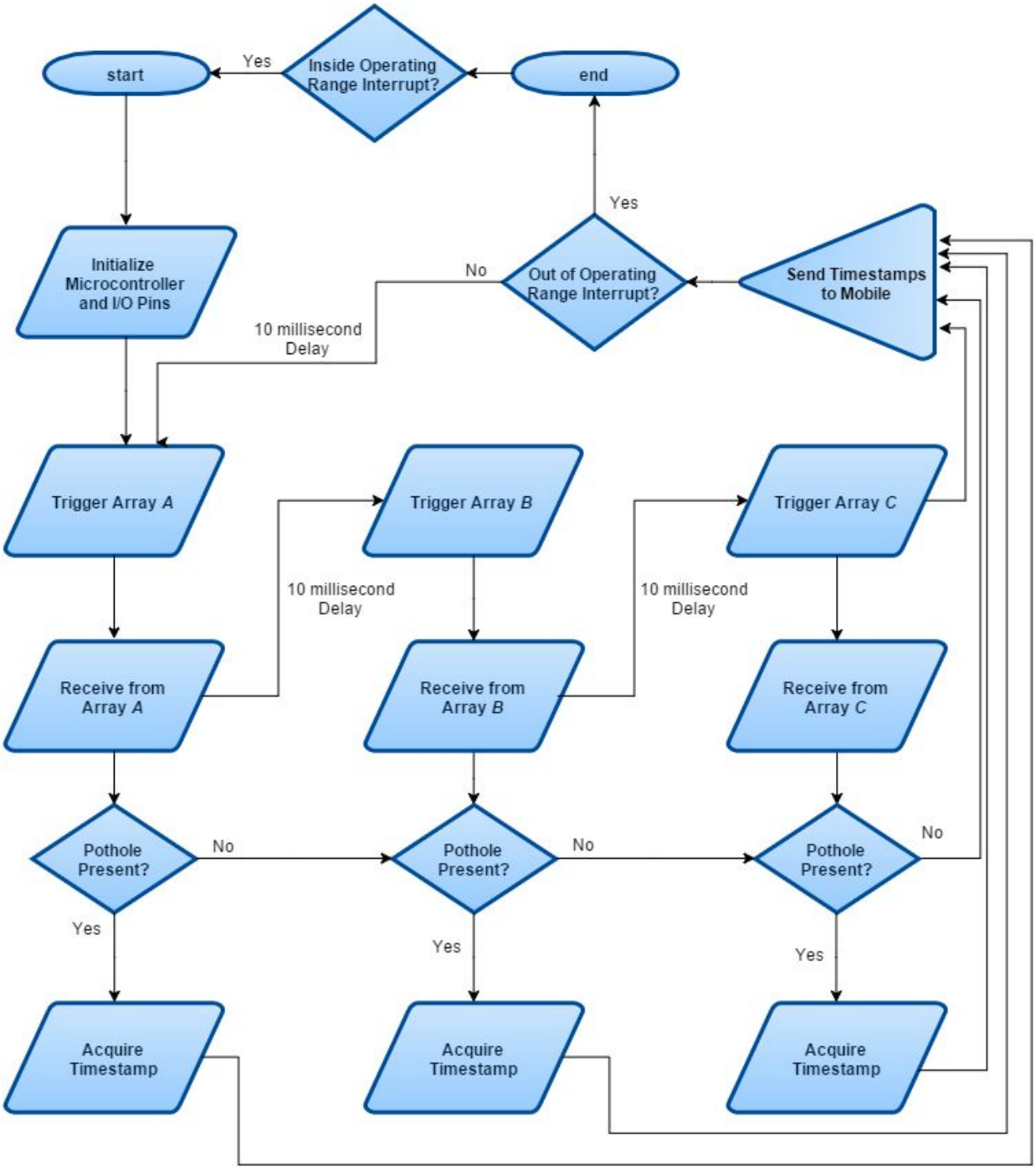

Figure 13: Firmware Flowchart

Before the microcontroller can begin to communicate with the sensor arrays in place on the bottom of the vehicle, it must initialize itself and the pins being used as either inputs or outputs. Most of the microcontroller initialization takes place in a

configuration header file, however the pins that get initialized as inputs and outputs will be configured in the Micro_Initialization function shown below.

Module	Micro_Initialization
Inputs	● None
Outputs	● None
Functionality	● Initialize the Microcontroller and set up the I/O pins and timers

Once initialized, the microcontroller will enter a superloop. The first thing that takes place in this loop is a trigger to an array of sensors via the Trigger_Sensor_Array function. This trigger tells the sensors in the array to send out their 40 kHz pulse, however listening for a response is not within the scope of this function.

Module	Trigger_Sensor_Array
Inputs	● Array to trigger
Outputs	● Trigger success bit
Functionality	● Send a 10 microsecond trigger to one of the sensor arrays

After triggering the sensors, the pins connected to the sensors' echo output will be watched to see when the echo is brought low again. If this pin is brought low in under 10 milliseconds, it can be determined that a pothole is present at that location.

Module	Receive_From_Sensors
Inputs	● Sensor array to listen to
Outputs	● Distance values from each sensor in the array
Functionality	● Fetches the distance values that each sensor outputs in an array

If it is determined that a pothole is present at the sensed location, a timestamp needs to be acquired from the RTC in order for the android application to determine to the best of its capabilities the location of the indicated pothole.

Module	Acquire_Timestamp
Inputs	● None
Outputs	● Current time
Functionality	● Acquires the current time when the function is called

After each sensor array is analyzed for potholes, there will be an option to send timestamps to the mobile device. This opportunity will only come once every few seconds, so as to reduce power consumption. If there are no timestamps to send, then this

function will not be called. However when there is at least one timestamp, they will be communicated via Bluetooth to a mobile device via this function.

Module	Send_To_Mobile
Inputs	● Sensor array to listen to
Outputs	● Distance values from each sensor in the array
Functionality	● Fetches the distance values that each sensor outputs in an array

If the system enters or leaves the operating range in either temperature or speed, a flag will be raised to check these parameters. When this flag is raised, the Parameters_Check function will go and see what the temperature and the speed of the vehicle is. If it is within operating range, the system will either start or continue operation. If it is outside the operating range, the system will be disabled.

Module	Parameters_Check
Inputs	● Bit determining if parameters need to be checked
Outputs	● Bit determining if system should be in operation
Functionality	● Checks the temperature sensor and speed data to see if the system is within the set operating range.

In addition to these main functions, there will be an Interrupt Service Routine (ISR) that will run whenever an interrupt is triggered via the temperature sensor or the android application's speed data. This ISR will update values correlating to the speed and temperature, which the Parameters_Check function utilizes, as well as the flag indicating that one or more of these parameters have changed.

Wireless Communications

A Laird BT900 Bluetooth module is used to establish wireless communications between the PIC microcontroller and the Android smartphone. While Wi-Fi was originally considered, it was eventually ruled out due to its high power consumption. The BT900 is a dual mode Bluetooth 4.0 BT and BLE module, BLE will be used for this application. While standard Bluetooth offers significantly higher data rates, BLE will be fast enough to handle all of this project's traffic in real-time, while using significantly less power. Per the BT900 datasheet, the module can draw 85mA while transmitting in Bluetooth mode, while only drawing 10mA in the equivalent BLE mode. The module offers SPI, I2C, and UART communications; SPI will likely be chosen for its speed and ease-of-use. The module also offers a U.FL antenna connector rather than a standard chip antenna; this allows a remote antenna to be brought up from beneath the vehicle if attenuation through the chassis interferes with communication.

As shown in the following calculations, it was determined that BLE's speed is more than adequate for this project.

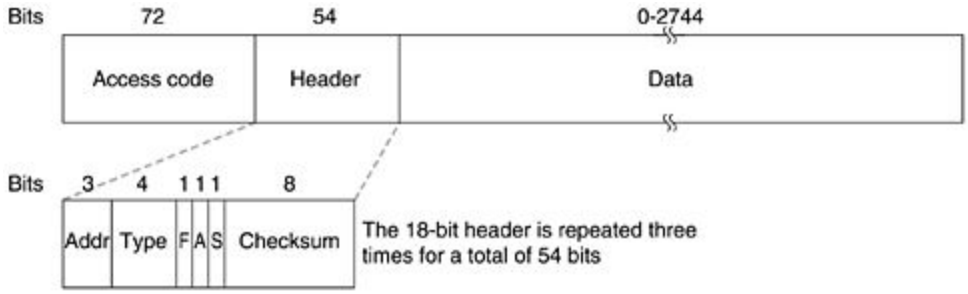

Figure 14: BLE Packet

Assuming a 20-character timestamp, and given 8 bits per ASCII character, each timestamp will be 20 bytes in length. This is significantly smaller than the maximum BLE payload size of 2744 bits. For the sake of discussion, assuming that the transmitted BLE packets have full payloads, and assuming that the sensing subsystem is detecting a pothole at a rate of one every 6.5ms (the maximum detection time), the required bandwidth of the BLE connection would be 422 kbps; this is well within the abilities of BLE, and should far exceed the actual realized bandwidth.

[SQ, AM]

Android Application

The android application will be utilized as the interface between the microcontroller and the cloud server database, as well as a way to display the pothole locations to the user after they have been logged into the system. It was chosen to build an Android application for this project, as the Android API fit well with the project's needs. The Android API includes a Bluetooth manager to allow easy pairing and communications with Bluetooth devices. The Google Maps API, built into the Android API, allows GPS coordinates to be easily acquired. The Google Maps API also provides a user interface which allows for users to scroll over and view location markers on an up to date Google Map.

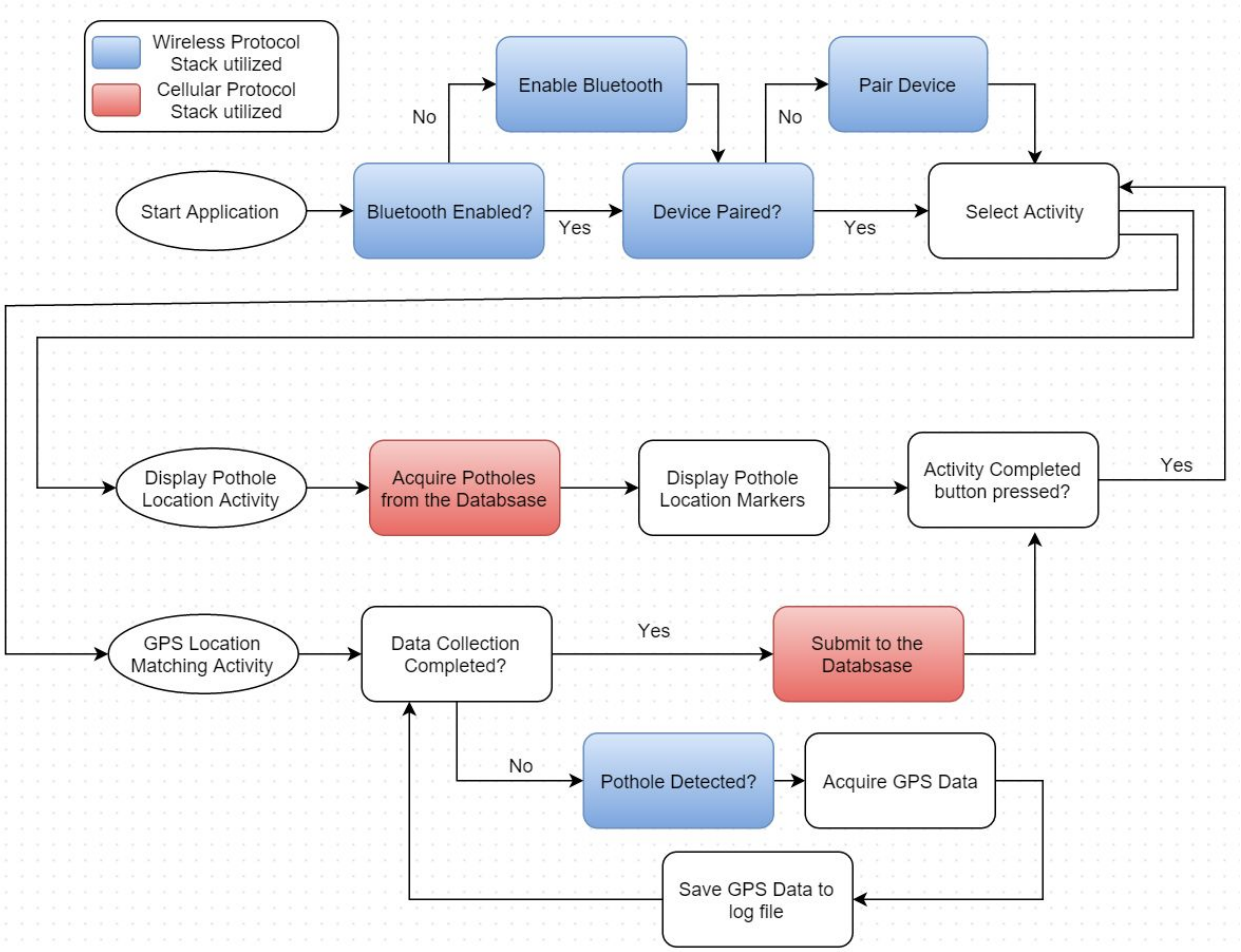

Figure 15: Android Application Flowchart

Two interface modules will need to be written to allow the Android application to communicate with the microcontroller and the cloud server. The Wireless Protocol Stack will handle the Bluetooth connection between the Android Application and the microcontroller, and will consist of all configurations of the Android API's Bluetooth manager. The module will handle the lower level connection of the link so that other parts of the app can easily send any information over the Bluetooth link without needing additional set-up. The Cellular Protocol Stack will be in charge of handling the interface between the cloud server database and the android application. All SQL queries will go through the Cellular Protocol Stack so the pothole information can be seamlessly stored in the database.

[AM]

Module	BluetoothController(Wireless Protocol Stack)
Inputs	• Bluetooth packet from the data processing subsystem's bluetooth module.
Outputs	• Timestamp to be utilized by the GPS Location Matching Activity

Functionality	● The Bluetooth Controller will be in charge of configuring bluetooth settings for the mobile device and creating the socket to allow communication between the data processing unit and the smartphone.

[AM]

The BluetoothController module will utilize Android Bluetooth Manager to set up bluetooth connections between the smart phone and the data processing system's bluetooth module. The Android class called the BluetoothAdapter will be used to acquire information about the bluetooth module built into the mobile device. The BluetoothAdapter will allow the user of the application to turn on the device's bluetooth communications so we can begin to look for devices to connect to. The BroadcastReciever android class is used to inform the user about the state of the device's bluetooth, meaning that the BroadcastReciever will let us know when bluetooth is turning on or off. These classes will be useful because users will not need to go into the settings of their device to manage the state of their bluetooth connection.

The next step for the Bluetooth Manager will be to pair the microcontroller with the smart phone. Before we can bond and connect the devices, we will need to have the devices discover each other using the BluetoothAdapter's startdiscovery() function. Once the devices are paired, the the BluetoothAdapter will automatically create a socket for both the mobile device and the microcontroller using the same RFCOMM. This will allow us to send and receive data from the mobile device to the microcontroller.

[AM]

Module	JSONParser
Inputs	● JSON strings from the output php files located in the cloud server.
Outputs	● A list of parsed pothole timestamps and location data to be displayed in the Display Pothole Location Activity.
Functionality	● The JSON parser will take the pothole information output by the php files located in the cloud server and parse them so that they can be used in the Display Potholes Location Activity.

[AM]

The JSON Parser class will be used to interpret the outputs of the http posts from the php files that will be stored in the server. The JSON parser will be able to retrieve and interpret the strings that are output from the php files located in the server so that they can be utilized by the application. The JSON parser along with the php files located in the server are the components that make up the Cellular protocol stack which is in charge of facilitating the transfer of data between the database and the android application.

Module	MainActivity
Inputs	● None
Outputs	● User will be directed to either the GPS Location Matching Activity and the Display Pothole Location Activity.

Functionality	● The MainActivity is simply a user interface that provide the users a way to to navigate between the GPS Location Matching Activity and the Display Pothole Location Activity.

[AM]

The MainActivity will provide the users with an interface that will allow them to navigate between the GPS Location Matching Activity and the Display Pothole Location Activity. This will be a very simple display with two buttons, one that will take the user to the GPS Location Matching Activity and one that will take the user to the Display Pothole Location Activity.

Module	Pothole class
Inputs	● None
Outputs	● None
Functionality	● The pothole class is a data structure that will hold all of the data about a single pothole.

[AM]

The pothole class will be created to aid us in organizing the all of the information that goes along with each pothole. The pothole class will have an object known as a "pothole". The member variables will be two doubles to hold the latitude and longitude of the pothole, and one string variable to hold the timestamp of the pothole. This class will also have setter and getter functions that will allow other parts of the code to easily access each of these member variables.

Module	GPS Location Matching Activity
Inputs	● GPS data ● Timestamp information from the microcontroller after it has passed through the BluetoothController.
Outputs	● GPS Speed Data to the MicroController. ● A "Pothole" data structure.
Functionality	● The GPS Location Matching Activity will associate GPS information with the pothole timestamp.

[AM]

The GPS Location Matching Program will receive the timestamp from the Wireless Protocol Stack which is made up of the BluetoothController class. The program will use the Android API to acquire the GPS coordinates of the detected pothole, and relay these coordinated to the Cellular Protocol Stack. This information will be stored in a "pothole" data structure to make it easier to work with the pothole information. Below in Example Code 3 is a sample of pseudocode that shows how the GPS Location Matching Program will function.

```
if (pothole_event_detected)
{
      gps_coords = get_current_gps_coordinates();
      send_to_server(gps_coords);
}
```

Sample Code 3: GPS Location Matching

Since acquiring GPS coordinates using the Android API will have some lag time, an interpolation algorithm will need to be written to more accurately find and log the correct coordinates of the potholes detected. The maximum specified speed for this project is 70 mph, which equates to 102.67 ft/sec. The update time for the Android GPS has been shown to be approximately 1.45 seconds, which would equate to one GPS update every 148.9 feet at 70 mph. This is significantly worse than the 30 feet accuracy that is specified in the project's engineering requirements. In order to bring the high-speed GPS accuracy in line with the project specification, interpolation will be used, as shown below.

The below pseudo code is an example of how we might calculate the change in meters from the current timestamp and the timestamp at which we actually detected a pothole.

[AM]

```
Meters_behind =(Current_Time  - Time_From_Microcontroller) *
Speed_In_meters/second;
```

Sample Code 4: Timestamp Calculation

This pseudocode is an example of how we might convert a change in position in meters to a change in position in latitude and longitude.

```
      lat = lat0 + (180/pi)*(dy/6378137);
      lon = lon0 + (180/pi)*(dx/6378137)/cos(lat0);
```

Sample Code 5: Latitude and Longitude Calculations

In the equation, lat0 and lon0 are the GPS coordinates for the location at which a pothole was detected, The variables dy and dx are the change in distance in meters from that position.

Figure 16: The Display Location Activity Screen

Module	Display Pothole Location Activity
Inputs	● Pothole data from the database passed in through the Cellular protocol stack.
Outputs	● A google map displayed on the screen showing the locations of all the potholes logged in the database
Functionality	● This Activity will display the potholes stored in the database on a Google Map.

[AM]

The Display Pothole Location Activity will utilize the Google Maps API to display all of the pothole locations stored in the server.The pothole information will be returned in the form of a "Pothole" data structure for ease of use. The following pseudocode shows the logic that will be used to display all of the pothole locations on the map for the user to review.

[AM]

```
pothole_array = get_potholes_from_server();
for (sizeof(pothole_array))
{
      place_map_marker(each_pothole);
}
```

Sample Code 6: User Review

[AM, SQ]

Cloud Server

The cloud server will house the database in which the list of pothole coordinates will be stored. In order to keep the database simple, there will be a single table in which all pothole coordinates and timestamps will be stored. SQL will be used to create and manage the database. The below diagram illustrates how the information will be stored in the database.

Table 4: Pothole Locations

Pothole #	TimeStamp	Latitude	Longitude
1	7:04:52	41.07155	-81.518357
2	8:35:33	37.722680	-77.987017
3	20:00:40	40.545703	-107.281050

[AM, SQ]

Along with the database, php files will be stored on the google cloud server. These php pages will be used to make http posts that will allow the android application to communicate with the database using SQL queries. An example of how the php file will query the SQL server when attempting to post a pothole to the database is shown below.

```
if (!empty($_POST))
{
      //initial query
      $query = "INSERT INTO suppliers(TimeStamp,Latitude,Longitude)VALUES(:TS,
      :lat, :long);
      ";
       //Update query
        $query_params = array(
        ':TS' => $_POST['TS'],
        ':lat' => $_POST['lat'],
        ':long' => $_POST['long'],
          );
              //execute query
          try {
              $stmt   = $db->prepare($query);
              $result = $stmt->execute($query_params);
           }
          catch (PDOException $ex) {
            $response["success"] = 0;
            $response["message"] = "Failed to run query: " .
            $ex->getMessage();
              die(json_encode($response));
          }
          $response["success"] = 1;
          $response["message"] = "Pothole successfully added!";
          echo json_encode($response);
      } else
      {
      ?>
                      <h1>Add Pothole Data</h1>
                      <form action="addpothole.php" method="post">
                          Pothole ID:<br />
                          <input type="text" name="id" placeholder="id" />
                          <br /><br />
                          TimeStamp:<br />
                          <input type="text" name="TS" placeholder="TS" />
                          <br /><br />
                          Latitude:<br />
                          <input type="text" name="lat" placeholder="lat" />
                          <br /><br />
                           Longitude:<br />
                          <input type="text" name="long" placeholder="long" />
                          <br /><br />

                         <input type="submit" value="Add Pothole" />
                      </form>
```

Sample Code 7: Example Php File

7. Level Diagrams & Explanations

7.1. System

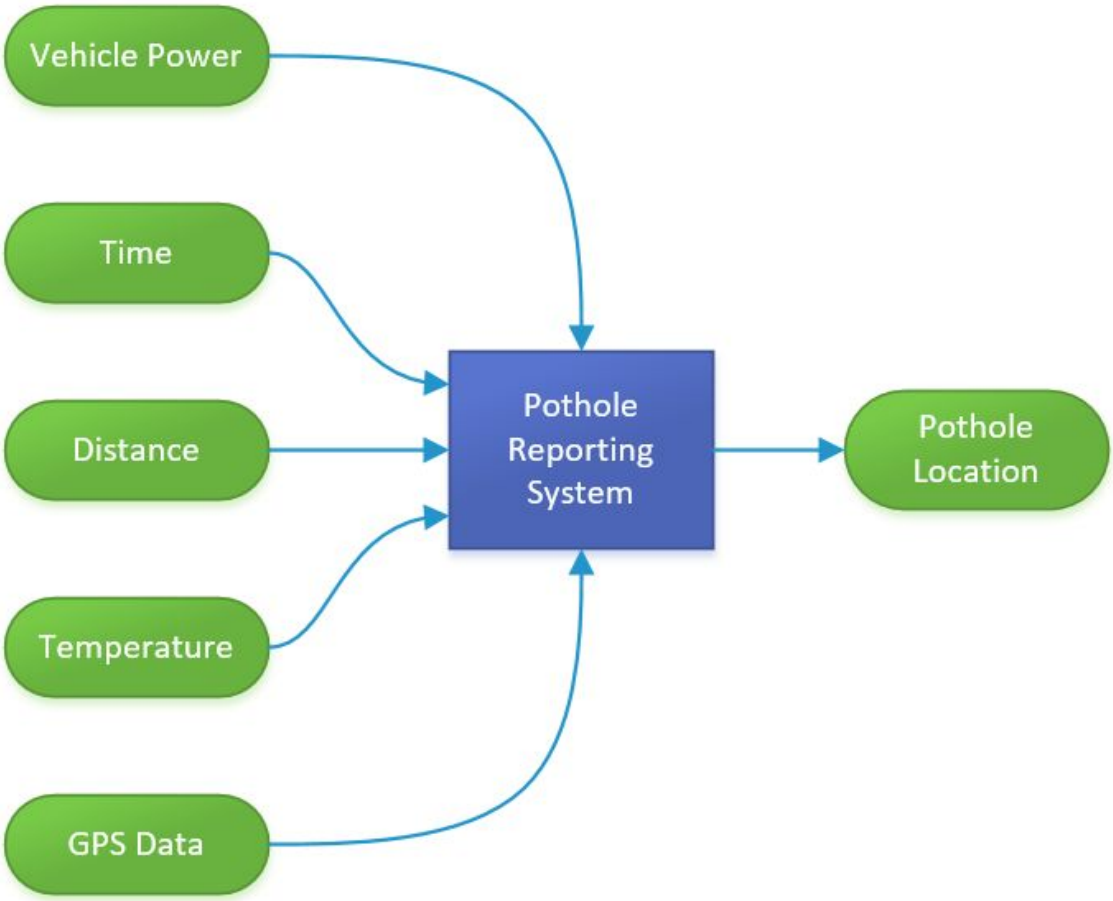

Figure 17: Level Zero Diagram

Module	Pothole Detecting System
Inputs	• Vehicle Power • Time • Distance • Temperature • GPS Data
Outputs	• Pothole Location
Functionality	• The module will detect potholes as well as obtain and log their GPS coordinates.

[BS, SQ]

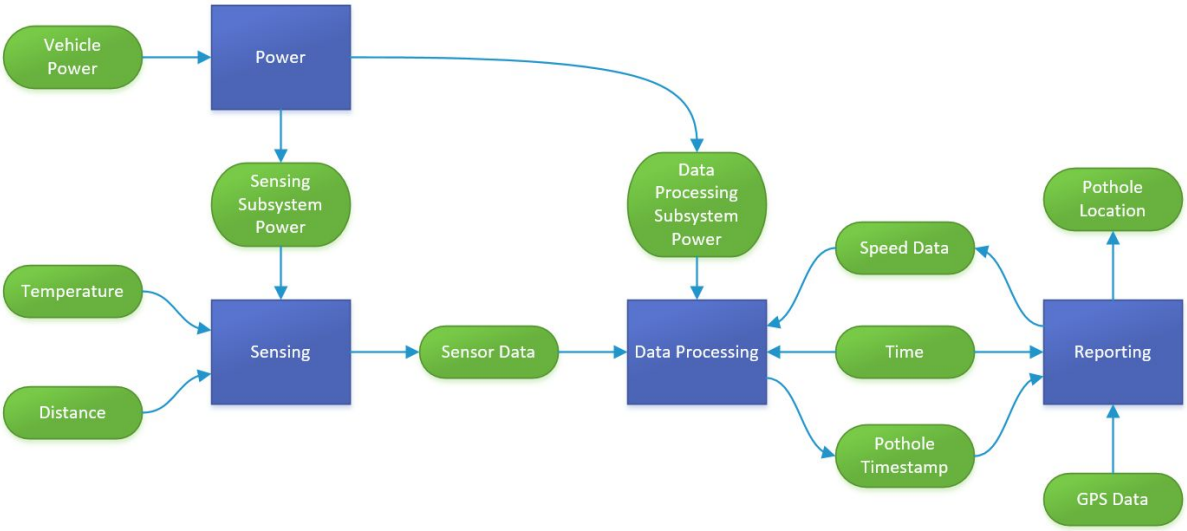

Figure 18: Level One Diagram

Module	Power
Inputs	• Vehicle Power
Outputs	• Sensing Subsystem Power • Data Processing Power
Functionality	• Converts vehicle electrical system power to clean, stable DC power for the sensing subsystem and data processing subsystem circuitry.

[SQ]

Module	Sensing
Inputs	• Data Processing Subsystem Power • Temperature • Distance
Outputs	• Sensor Data
Functionality	• Enables the system if the temperature is within the operating range and reports distance data for further processing.

[EH]

Module	Data Processing
Inputs	• Sensor Data • Data Processing Subsystem Power • Time • Speed Data
Outputs	• Pothole • Timestamp

Functionality	• Processes the data from the sensors and determines if a pothole is present. If so, sends a flag to the mobile device that a pothole is present.

[BS]

Module	Reporting
Inputs	• Time • Pothole Timestamp • GPS Data
Outputs	• Speed Data • Pothole Locations
Functionality	• When the reporting system receives a notification from the data processing system that a pothole has been detected, the reporting system will acquire the GPS coordinates for the detected pothole and log the information to the server. The reporting system will also display pothole locations logged in the server for user review.

[AM]

7.2. Hardware

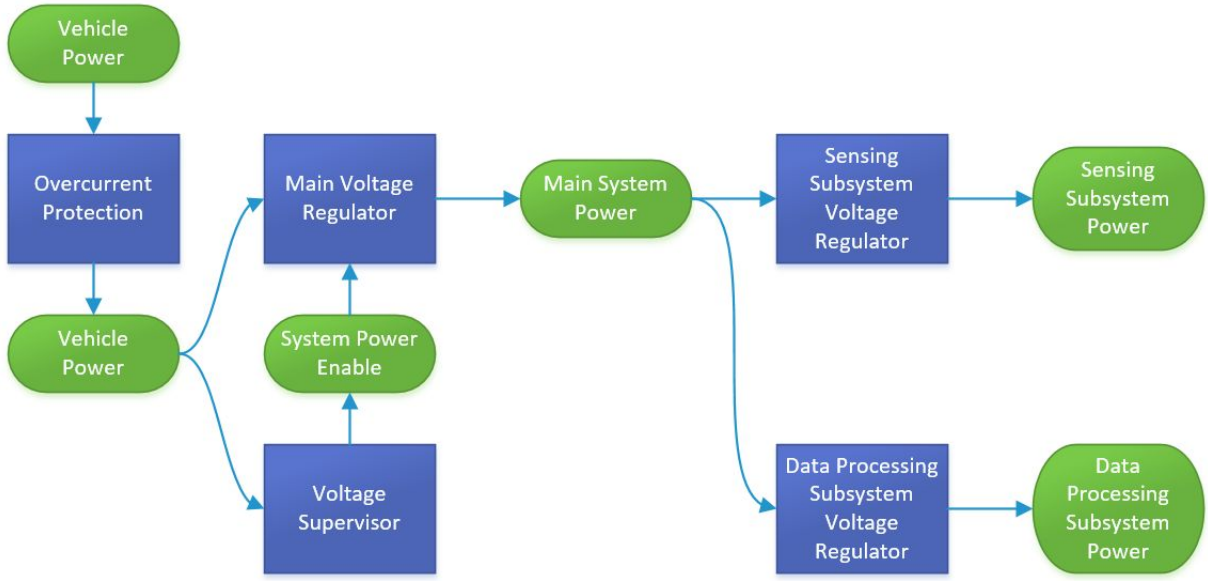

Figure 19: Level Two Diagram: Power

Module	Overcurrent Protection
Inputs	• Vehicle Power
Outputs	• Vehicle Power
Functionality	• Protects the vehicle battery from a short circuit in the pothole reporting system circuitry

Module	Voltage Supervisor
Inputs	• Vehicle Power
Outputs	• System Power Enable
Functionality	• Enables or disables the main voltage regulator based on the vehicle power system voltage level.

Module	Main Voltage Regulator
Inputs	• Vehicle Power • System Power Enable
Outputs	• Main System Power
Functionality	• When enabled, generates a clean DC voltage for the pothole reporting system electronics.

Module	Sensing Subsystem Voltage Regulator
Inputs	• Main System Power
Outputs	• Sensing Subsystem Power
Functionality	• Converts the main system DC voltage level to a DC voltage level appropriate for use with the sensing subsystem.

Module	Data Processing Subsystem Voltage Regulator
Inputs	• Vehicle Power
Outputs	• Data Processing Subsystem Power
Functionality	• Converts the main system DC voltage level to a DC voltage level appropriate for use with the data processing subsystem.

[SQ]

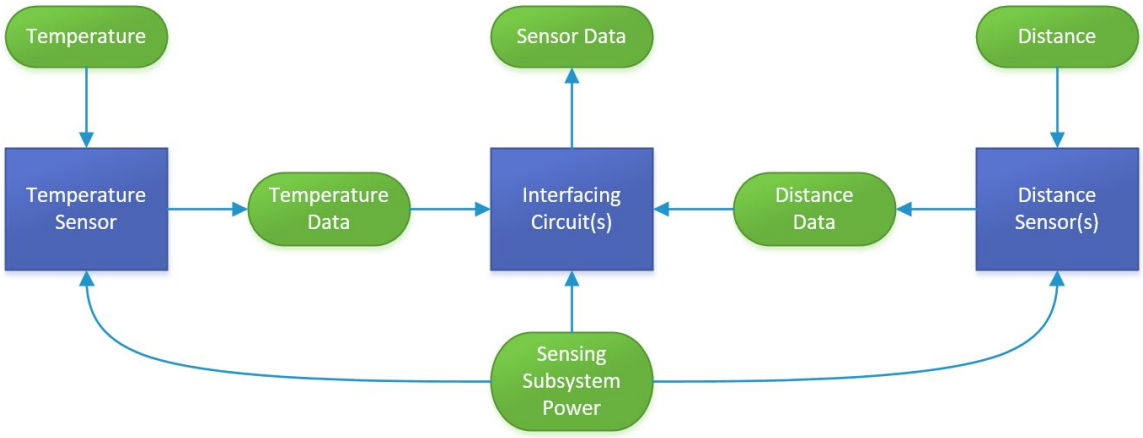

Figure 20: Level Two Diagram: Sensors

Module	Temperature Sensing
Inputs	• Sensing Subsystem Power • Temperature
Outputs	• Temperature Data
Functionality	• Gathers external temperature to transfer to interfacing circuitry.

Module	Distance Sensing
Inputs	• Sensing Subsystem Power • Distance
Outputs	• Distance Data
Functionality	• Reads distance to ground, pothole bottom, or other road anomalies to transfer to interfacing circuitry.

Module	Interfacing Circuitry
Inputs	• Sensing Subsystem Power • Temperature Data • Distance Data
Outputs	• Sensor Data
Functionality	• Enables distance and temperature data to be converted (if necessary) and reported to microcontroller. Temperature enables the system and distance allows for the reporting of potholes.

[EH]

7.3. Firmware

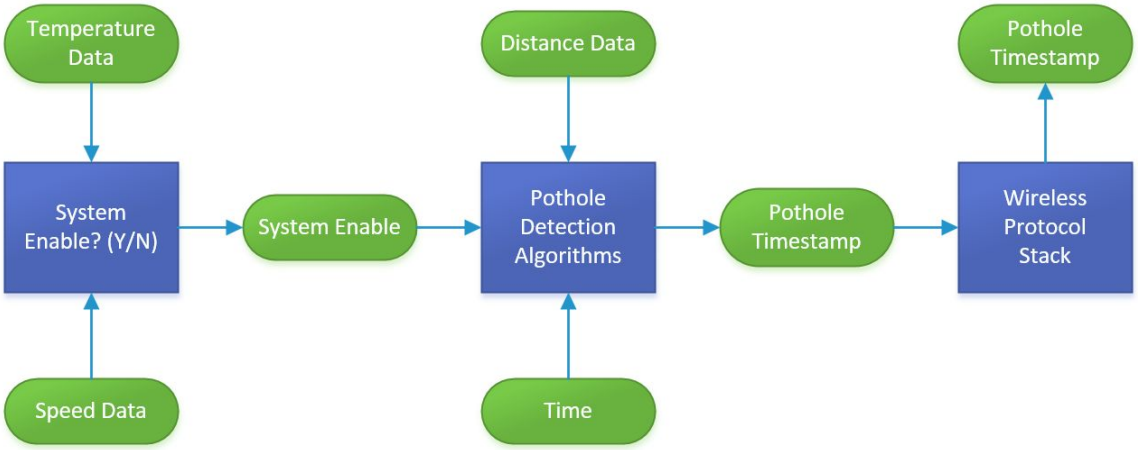

Figure 21: Level Two Diagram: Firmware

Module	System Enable
Inputs	• Phone GPS Speed Data • Temperature Sensor Data
Outputs	• System Enable
Functionality	• Determine if the speed of the vehicle and climate outside are within the operating parameters of the system

Module	Pothole Detection Algorithms
Inputs	• System Enable • Distance Data • Time
Outputs	• Pothole Flag • Timestamp
Functionality	• Analyze the data from the distance sensors to determine if a pothole has been seen, and create the flag to be sent to the mobile device.

Module	Wireless Protocol Stack
Inputs	• Pothole Flag • Timestamp
Outputs	• Pothole Flag • Timestamp
Functionality	• Communicate Via BLE to send Pothole flags to the mobile device.

[BS]

7.4. Software

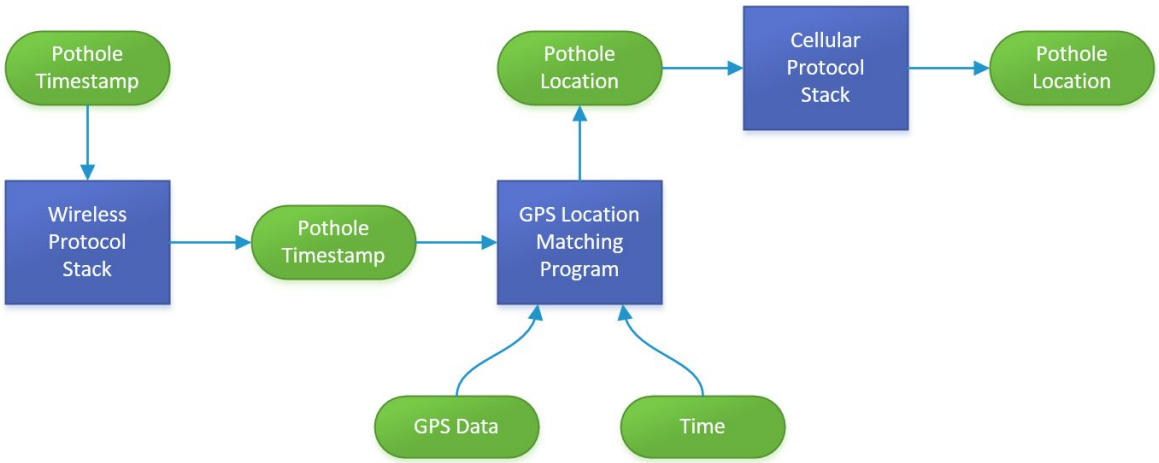

Figure 22: Level Two Diagram: Software

Module	Wireless Protocol Stack
Inputs	• Pothole Timestamp
Outputs	• Pothole Timestamp
Functionality	• The Wireless Protocol Stack will interpret the data received through the Bluetooth connection and convey the pothole timestamp data to the GPS Location and Matching Program.

Module	GPS Location Matching Program
Inputs	• Pothole Timestamp • Time • GPS Data
Outputs	• Pothole location
Functionality	• The GPS Location Matching Program will receive the timestamp from the Wireless Protocol Stack. The program will use the built in GPS data from the smartphone to acquire the GPS coordinates of the detected pothole, and relay these coordinated to the Cellular Protocol Stack.

Module	Cellular Protocol Stack
Inputs	• Pothole Location

Outputs	• Pothole Location
Functionality	• The Cellular Protocol Stack will take the coordinates from the GPS Location Matching Program and upload the data to the database stored in the Cloud Server.

[AM]

8. **Tentative Parts List**

Part No.	Description	Quantity
PIC24EP64GP206	Microcontroller	1
MCP79510	Real-Time Clock	1
HY-SRF05	Ultrasonic Sensor	1
MCP9808	Temperature Sensor	1
N/A	PCB	1
1709678	20-position Screw Terminal Block	2
NCV33161DMR2G	Voltage Supervisor	1
LM22676	Switching Voltage Regulator	1
NCV1117DT33T5G	Linear Voltage Regulator	1
GRM188R71C104KA01D	.1uF 10-20V Capacitor, Ceramic	1
GRM188R61A105MA61D	1uF 10V Capacitor, Ceramic	1
T491A106K016AT	10uF 16V Capacitor, Tantalum	1
ERJ-6GEYJ103V	10kΩ Resistor	2
ERJ-8GEYJ471V	470Ω Resistor	1
	4A Fuse	1
SBR1U400P1	400V 1A Diode	1
MURS340HE3	400V 4A Diode	1
5KASMC24AHM3/57	35.5V Schottky Diode	2
EGXL500ELL221MJ20S	220uF Ceramic Capacitor	1
ERA-6AEB2493V	249kOhm Resistor	1
ERA-6AEB2672V	26.7kOhm Resistor	1
BUK9209-40B,118	P-channel Enhancement Mosfet	1
C5750X7S2A106M	10uF Capacitor	1
GRM21BR71H105KA12L	1uF Capacitor	1
C0805C103K5RACTU	10nF Capacitor	1
ELL-6UH151M	150uH Inductor	1
PMEG6010CEH,115	60V,1A Schottky Diode	1
GRM21BR61A106KE19L	10uF Capacitor	3
01550900M	Fuse Holder	1

9. **Design Team Information**

Elizabeth Hammell, Hardware Lead
Electrical Engineering
Relevant Coursework: Control Systems I & II, Computer Systems, Digital Signal Processing, Embedded Systems Interfacing, Sensors and Actuators

Alissa McGill, Software Lead
Computer Engineering
Relevant Coursework: Data Structures, VLSI Circuits and Systems, Analog IC Circuits, Applied Numerical Methods I

Sean Querry, Team Leader
Electrical Engineering
Relevant Coursework: Computer Systems, Digital Communication, Wireless Communication, Antenna Theory, Electromagnetic Compatibility, Embedded Systems Interfacing, Digital Signal Processing

Brian Simmons, Archivist
Computer Engineering
Relevant Coursework: VLSI Circuits and Systems, Analog IC Circuits, Active Circuits, Digital Signal Processing, Applied Numerical Methods I & II

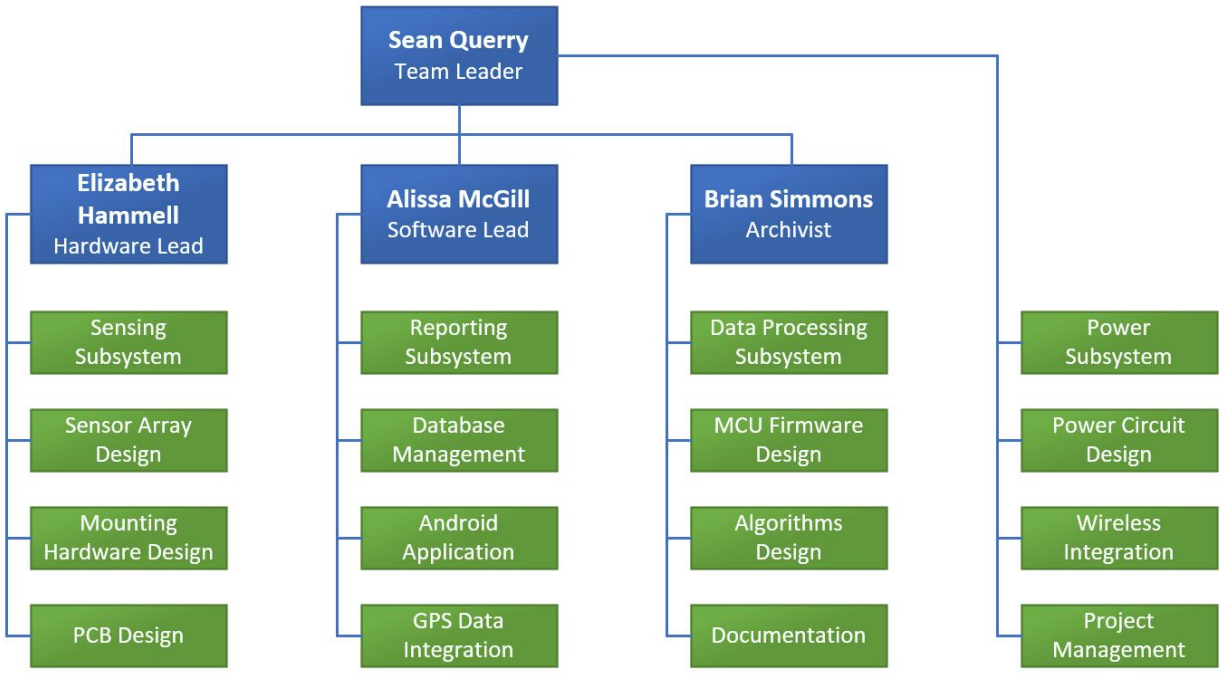

Figure 23: Organizational Chart

[AM, BS, SQ, EH]

10. References

[1] J. Cronin, "System and method for sensing and managing pothole location and pothole characteristics," US 20140196529 A1, Jul 17, 2014.

[2] J. Lemelson, “For repairing road surfaces,” US 5294210 A, Mar 15, 1994.

[3] J Bridgers, “Mobile pothole detection system and method,” US 20140355839 A1, Dec 4, 2014.

[4] X. Yu and E. Salari, “Pavement Pothole Detection and Severity Measurement Using Laser Imaging”,Department of Electrical Engineering and Computer Science:The University of Toledo.

[5] Sandeep Venkatesh, Abhiram E ,Rajarajeswari S, Sunil Kumar K M and Shreyas Balakuntalay
“An Intelligent System to Detect, Avoid and Maintain Potholes: A Graph Theoretic Approach”, 2014 Seventh International Conference on Mobile Computing and Ubiquitous Networking (ICMU)

[6] Eric S. Li ,”Physical Optics Models for the Backscatter Response of Road-Surface Faults and Roadside Pebbles at Millimeter-Wave Frequencies”, IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, VOL. 51, NO. 10, OCTOBER 2003

[7] Harish Anil Jamakhandi,Srinivasa K.G, “ INTERNET OF THINGS BASED REAL TIME MAPPING OF ROAD IRREGULARITIES”,Assistive Devices Group
M. S. Ramaiah Institute of Technology, Proceedings of International Conference on Circuits, Communication, Control and Computing (I4C 2014)

[8] Littelfuse, Inc., “Suppression of Transients in an Automotive Environment,” AN9312.5 application note, Jul. 1999.

[9] *Road Vehicles - Electrical disturbances from conduction and coupling - Part 2: Electrical transient conduction along supply lines only*, ISO Standard 7637-2, 2004.

[AM, BS, SQ, EH]