Pothole Tracker

Bill Quigg, EE, Daniel Chin, CSE, Mike Catalano, EE, and Muhammad Mir, CSE

***Abstract*—The Pothole Tracker is a device that takes pictures of potholes in a moving vehicle using two cameras while marking their location using a GPS and measuring their size and depth with either image processing or accelerometers. The system is initiated by the press of a button to record information regarding the pothole and send data to the database. All the data gathered from the devices will be stored in a database. A website connects to the database to display all the information regarding the potholes allowing Public Works Departments to view the data and make necessary repairs to the roads.**

I. Introduction

POTHOLES are holes that form in the pavement from excessive use or by extreme weather. They are a nuisance and a hazard to drivers. These craters cause serious damage to cars resulting in costly repairs. According to AAA, potholes cost drivers $6.4 billion a year [10]. This is around $2,000 throughout the life span of a car just from road conditions. These problems in the road are costing drivers their money, with potholes being at the forefront of the cause forcing them to demand their efficient repair.

The first step in repairing potholes is to know the location of the pothole. Not all potholes are big enough in size and depth to cause damage and certainly not every pothole can be repaired. How do we keep track of potholes and their level of repair?

Most places with a large population, such as cities, have a database of potholes that keep track of their location, size, and status of repair. The city of Seattle, Washington implements a similar system but with a key difference in how the database acquires and stores the information. The citizens of Seattle submit forms reporting potholes, which are reviewed and put into the database [8]. Although this method is useful, manually storing the information into a database is a hassle. With the advancement of technology, there may be better ways to keep track of potholes.

An efficient solution to this problem would be a system that can track potholes and submit the required data into a database. Regardless of the technology, the system will need to go out into the field and analyze the potholes first hand.

The device needs to be compact enough to fit into a vehicle and not be in the way of anything. Due to this constraint we decided that the length, width, and height would have to be under 6in. In addition it needs to be able to be powered by the car itself therefore the system must use less than 5V. It has to transfer data so wireless and network capabilities are a must. There will also need to be driver input to tell the device when to start analyzing the pothole.

Specification	Value
Power	<5V
Height	<6in
Width	<6in
Length	<6in

II. Design

A. Overview

Our solution to the problem is to take pictures of potholes and use image processing to extract important information, such as size, depth, and location, and store it into a database. The block diagram in Figure 1 shows the solution in the form of a block diagram. We will use Raspberry Pi [7] boards for our system because they are compact powerful microcomputers. They are capable and affordable. There will be a total of four boards that will make up the main system. Two will house the cameras, one will be the Network Attached Storage (NAS), and one will be used for solely image processing. Due to the complexity of the problem at hand and the variation in driving speeds, we have to put a constraint on the speed at which the driver can drive at. The constraint is there to allow the camera to get good footage of the potholes that the driver wants to document. The test environment is a residential area with a speed limit of thirty miles per hour. When the driver sees a pothole coming up from at least 50ft away, they press a button that activates two cameras. The two cameras on separate boards record a video of the road with the pothole in at least a few frames. The boards then transmit the two videos to the NAS to be stored along with a GPS location and accelerometer readings. The accelerometers are storing data anytime the car drives over a pothole. The image processing board then grabs the videos from the NAS and processes them to figure out the size and depth of the pothole. It will then transmit the processed image and information to the NAS. At this point the NAS sends the data to a database to be stored. A webpage will access the database and display the locations of potholes on a map.

M. Catalano from Abington, Ma (mcatalan@umass.edu).
D. Chin from Stoughton, Ma (dhchin@umass.edu).
B. Quigg from Pembroke, Ma (wquigg@umass.edu).
M. Mir from Stoughton, Ma (mmir@umass.edu).

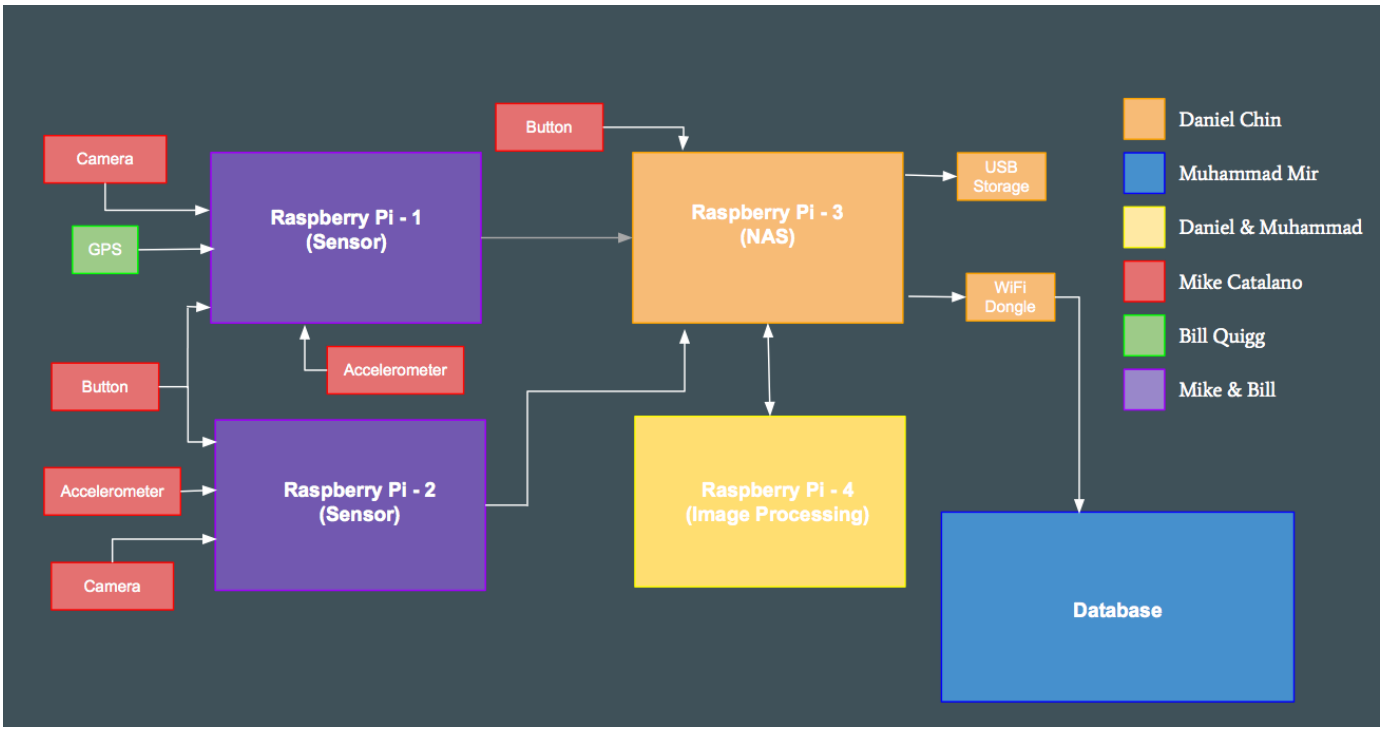

Figure 1: Block Diagram

B. Sensing

Incorporating a button into a system requires additional hardware that ensures reliable switching. Switch bouncing is a problem that degrades the integrity of a mechanical switch. A switch bounce occurs when a mechanical switch opens and closes. When this happens, unnoticeable vibrations on the contacts within the switch vibrate. This vibration causes the switch to open and close several times, typically within a period of a few hundred microseconds. An example of a switch bounce is shown in Figure 2, where the switch bounces twice before settling to a logical High. Switch debouncing in digital circuitry is very important when applications require precise timing. If a bounce were to occur, the Raspberry Pi would interpret the faulty input as multiple button pushes. There exists both software and hardware solutions to the switch bouncing issue. For the purposes of this project, hardware will be used. This decision was made based on the need for conserving processor resources for the image processing. Image processing will consume considerably more resources than switch debouncing, so eliminating the need for software is desirable whenever possible.

The circuit chosen to settle the switch bouncing is an RC debouncer that utilizes a Schmitt Trigger. The schematic for this circuit is shown in Figure 4. The circuit operates as following. When the button is not pressed, the capacitor charges through the series combination of R_1 and R_2 to a voltage close to V_{cc}. The voltage across the capacitor is the input voltage to the Schmitt Trigger inverter. When the capacitor is charged to Vcc, the output of the Schmitt Trigger will output 0V, a logical Low. Alternatively, when the button is pressed after charging for some time, the capacitor will discharge through R_2. This causes the voltage across the capacitor to go to 0V. If the voltage across the capacitor is 0V, the Schmitt Trigger will output V_{cc}, a logical High.

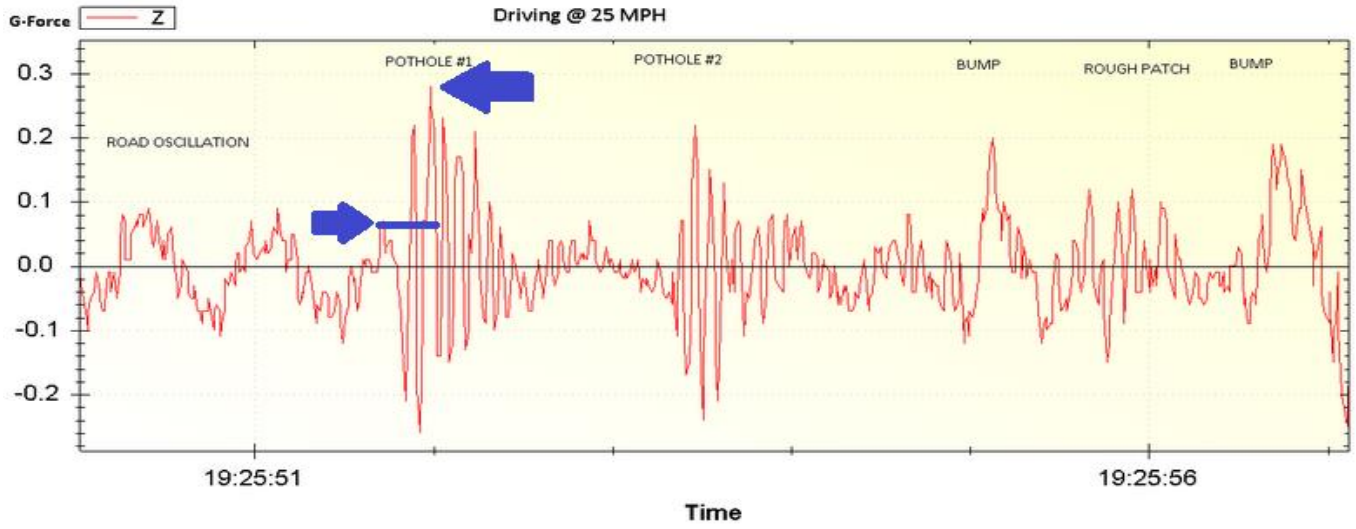

Figure 2: Accelerometer Reading

The design of this circuit focuses on maintaining a certain voltage long enough to settle any bouncing. The equation that dictates the voltage across a charging capacitor as a function of time is,

$$V_{cap} = V_{final}(1 - e^{-t/RC})$$

The previous equation can be rearranged in order to solve for R,

$$R = -t/C*\ln(1 - V_{cap}/V_{final})$$

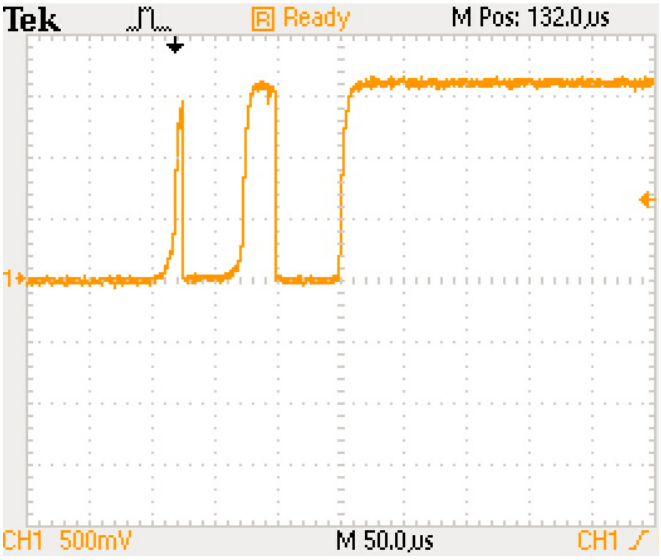

Figure 3: Before

Choosing a value of 0.1µF for the capacitor, C, is the first step in determining values for R_1 and R_2. The equation accounts for resistance in the exponent as R, which is equivalent to $R_1 + R_2$. V_{cap} is the voltage across the capacitor. The Schmitt Trigger will switch from a logical Low to High when the voltage across the capacitor goes above 0.8V. Therefore, $V_{cap} = 0.8V$. V_{final} is the voltage the capacitor is charging to, which is V_{cc}. For this project, $V_{cc} = 5V$. Finally, the time required to settle the bouncing comes from Figure 3, where the switch settles in approximately 175µs. This is the worst-case time recorded from numerous tests. Substituting these values into the previous equation yields $R = R_1 + R_2 = 10037\Omega$, which will be rounded to the closest standard resistance value of 10kΩ.

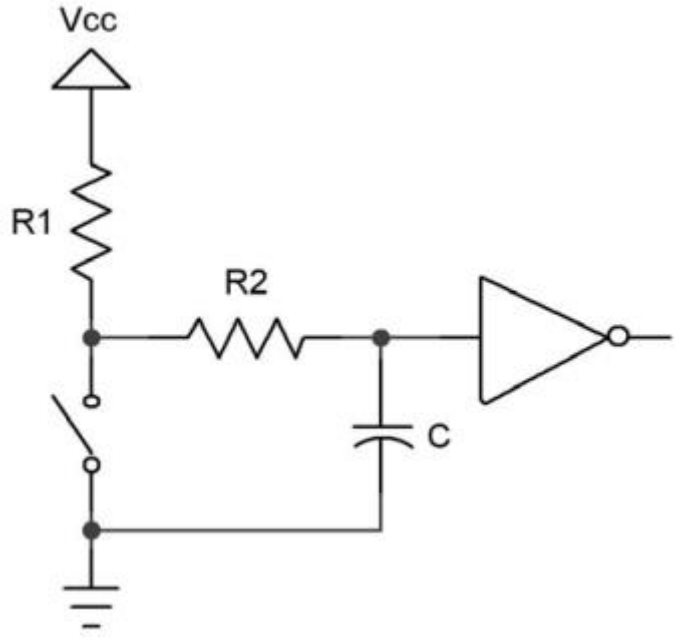

Figure 4: Circuit

Another expression for R is needed in order to find values for R_1 and R_2. Consider the following equation for a capacitor discharging,

$$V_{cap} = V_{initial}(e^{-t/RC})$$

This equation can be rearranged as an expression for R,

$$R = -t/C*\ln(V_{cap}/V_{initial})$$

V_{cap} assumes a value of 1.6V, the voltage at which the Schmitt Trigger will switch from a logical High to Low. $V_{initial}$ is equivalent to V_{cc} (5V). The values for C and t used in the previous calculation remain the same in this calculation as well. Substituting these values into the previous equation for R yields $R = R_2 = 1536\Omega$. The value for R_2 will be rounded to 1.5kΩ.

There is now a system of two equations in order to solve for R_1 and R_2,

$$R1 + R2 = 10k\Omega$$
$$R2 = 1.5k\Omega$$

Solving for R1 yields a value of 8.5kΩ, which is rounded to 8.3kΩ. The final design involves the following components, R1 = 8.3kΩ, R2 = 1.5kΩ, C = 0.1µF, and an SN74LS14N Schmitt Trigger will be used. The following graphic shows the output of the switch debouncing circuit. Note how the voltage rises sharply without any bouncing. This is the desired effect.

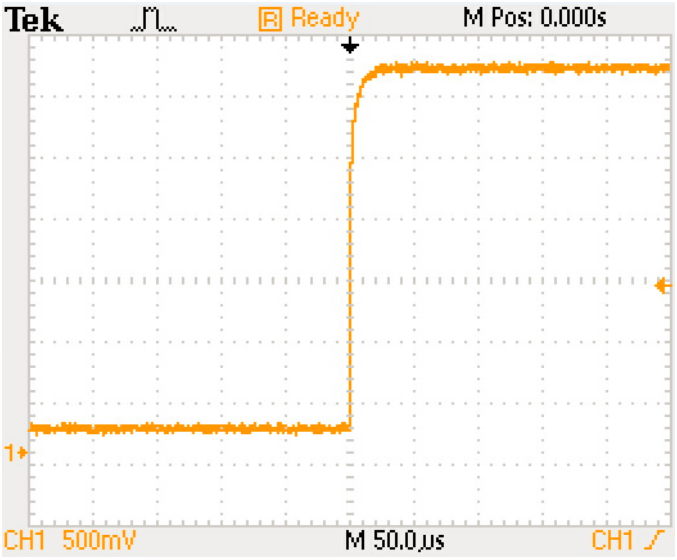

Figure 5: After

C. Image Processing

There were many other technologies we considered aside from image processing. Lasers were a promising idea but obtaining one was expensive and out of an efficient budget. Lasers would have been able to perform a 3D reconstruction of the pothole, which would allow us to manipulate it and perform the necessary calculations. There was also an issue of generating multiple lasers or rotating one laser across the road. The complications lead us to find a more simple method. We also considered using sonar over

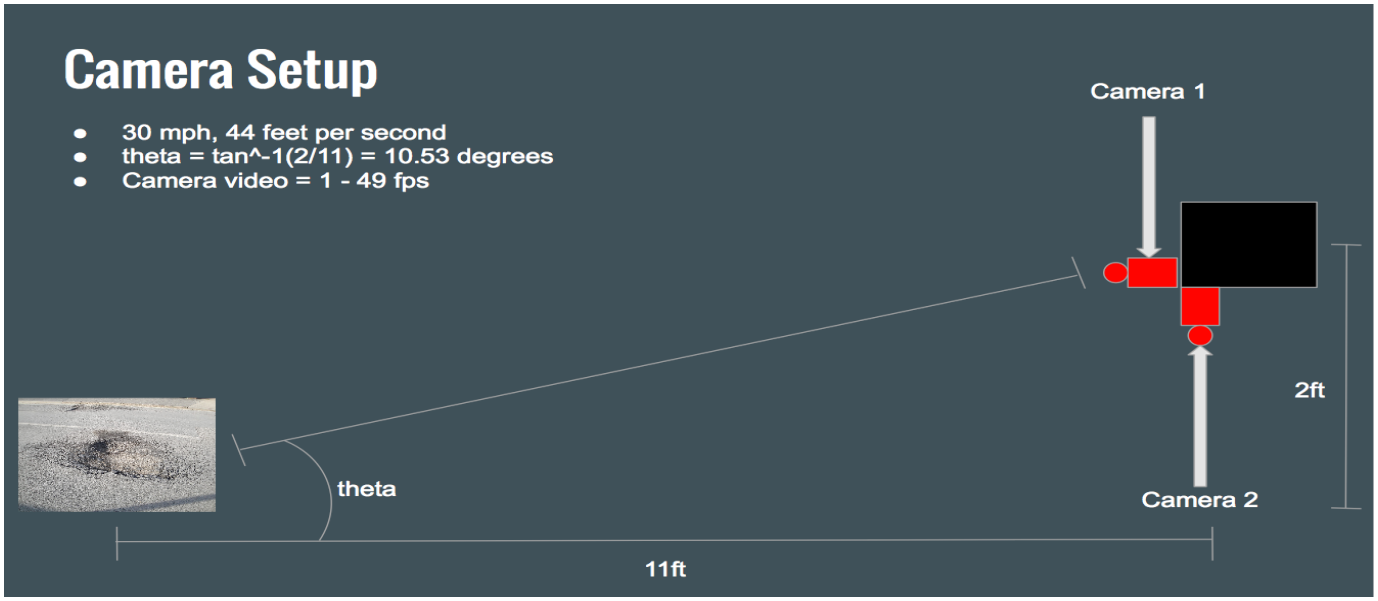

Figure 6: Camera Setup

the use of a button input to allow the system to automatically detect when a pothole was approaching. Unfortunately the available sonars for Raspberry Pis [6] had a limited range of fifteen feet restricting the amount of time to process and record from the cameras.

D. Block 1: Raspberry Pi 1

The first Raspberry Pi board (see Figure 1) will be equipped with a camera [4], GPS [2], and accelerometer [1]. The system activates with the press of a button, which is also connected to the second Raspberry Pi board. The camera will be at an angle of 11° giving it a good view of the pothole from eleven feet away. The video from this camera will be used to measure the depth of the pothole. This camera will be mounted on the front bumper of the car, which will be around two feet above the ground.

A method for determining the depth of pothole that will be explored in this project is through the use of accelerometers. Accelerometers are sensors that collect data regarding force. In the context of the Pothole Tracker, the accelerometers will measure the force sustained by the car when driving over a pothole. Two accelerometers will be mounted symmetrically to the vehicle's car frame in order to measure the force sustained by the individual front tires. Mounting the accelerometers to a section of the car that remains static during impact is critical to collecting useful data. If the accelerometers were mounted to a dampened section of the car's suspension, the resulting data would correspond to a pothole that is shallower than it really is.

The basis for calculating the depth of a pothole comes from the kinematic equation for free-fall. When a car drives over a pothole, the front tire of the car is briefly in free-fall. The equation for free-fall in the z-direction is,

$$\Delta z = 0.5at^2$$

Where Δz is the change in position in the z-direction, a is the acceleration recorded by the accelerometers, and t is the time period over which the car is free-falling into the pothole. Figure 2 will provide an example of this calculation. The period of time over which the front tire is in free fall is enclosed by the blue arrows. This interval is approximately 0.25s. The maximum acceleration experienced by the accelerometer is approximately 0.28g, where g is the acceleration due to gravity ($9.81 m/s^2$). The maximum acceleration corresponds to the vehicle's tire making contact with the bottom of the pothole. Substituting these values for acceleration and time into the equation for vertical displacement yields a value of 0.086m or 3.4in. The actual depth of the pothole is 4in.

In order for more accurate results to be calculated, the accelerometer data must be filtered. A bandpass filter should be designed such that low frequency anomalies as well as high frequency noise gets rejected. Low frequency anomalies correspond to the closing of a car door or a passenger moving around within the car. High frequency noise is the result of the car's engine vibrating. Based upon the previous graph, the frequency that corresponds to the car free-falling is 12Hz. This frequency can be thought of as a "pothole frequency", where this is a value included within the pass-band of the filter. Further testing will be done in order to find a sufficient range of pothole frequencies.

The GPS will always be on, similar to the accelerometer. It will be polling the longitude and latitude of the car from the GPS satellites 10 times every second The reason one reading every second is viable is because the car is moving at thirty miles an hour, which is forty-four feet a second, and the coordinates do not need to be the exact location of the pothole but rather a relative location. When the driver presses the button or the accelerometer detects a pothole, the Raspberry Pi board will grab the latest location from the GPS device. The GPS will be mounted near the location of the board it is connected to, which should allow for good GPS Satellite reception.

All three systems will be synced and the data each system gathers will be time stamped to allow organization and grouping. After acquiring the data, all the information such as

the video, GPS coordinates, and accelerometer readings, will be sent to the third Raspberry Pi board, which is the NAS.

The components that are connected to the Raspberry Pi were designed to be integrated with the board. Apart from the camera, all the devices need to be wired separately to one or more GPIO header pins in order to function properly. Some of the wiring will be completed on a breadboard to make the wires easier to manage. Each component will be coded with Python.

As we continue working on the project, we will need to learn how to program in Python to interface with the devices. For the accelerometer, we will need to experiment the detection range of the potholes in order to filter out unwanted data. We will also need to experiment with the location of the accelerometer in order to get the best reading when the car goes into a pothole.

E. Block 2: Raspberry Pi 2

The second Raspberry Pi board is connected to the same button as the first board. This board will be equipped with another camera, and an accelerometer. The camera on this board will be at an angle of 90°, facing it directly towards the ground. The video recorded from the camera will be used to measure the size of the pothole, specifically its diameter. The camera will be mounted on the front bumper, giving it a clear view of the ground while allowing it to capture an image as wide as the car.

The accelerometer will work exactly like the one attached to the first Raspberry Pi board. It will continuously get readings and will be filtered to only detect potholes. The location to mount the device will also be similar to the first, but will go on the opposite side of the car to get more coverage in the event the other wheel is steered into a pothole.

Since the first and second Raspberry Pi boards are operated using the same button, both boards will need to be synced to each other. The first camera will begin recording at around half a second after the button is pressed while the second camera will record a full second after the button is pressed. All the videos and data will need to be timestamped with the appropriate information in order to be recognized as a group for when the data is organized and processed. Finding a way to sync all the devices and their data will be something we need to learn how to do.

F. Block 3: Network Attached Storage

The third Raspberry Pi board will be dedicated to become a Network Attached Storage for the first two boards, where they capture and send the data, and the fourth board, which will be grabbing the videos from the NAS, processing them, and finally sending them back to the storage. The NAS will be connected to the other boards with either USB or Ethernet cables. Both USB and Ethernet will have the issue of dealing with a limited bandwidth while a high amount of data will be streamed from all three boards, though not at the same time. Also both methods will require some form of switching amongst the three boards in order to send or receive data.

The board will be connected to a storage device using one of its USB ports. This storage will either be a USB flash drive or a larger external drive, depending on how the data will be stored in the NAS. The initial video that the NAS will receive will be large but after it is processed, it will only contain the single best frame of the pothole along with the processed image.

The data flow is relatively straight forward and only in one case does data have to be sent and received. Raspberry Pi boards one and two will only be sending data to the NAS for storage. The image processing Raspberry Pi, the fourth board, will be requesting videos from the NAS to be sent to it and after it has been processed, it will be sent back along with information about the depth. Finally the last path of data will be sent from the NAS to the database.

Since the database is a separate system from the four Raspberry Pi boards, an internet connection is required to send data to it. To allow the board to access the internet through Wi-Fi, a USB Wi-Fi dongle will be connected in one of the USB ports. This allows the board to access a network connection without needing an Ethernet cable.

The bandwidth between the USB ports and the Ethernet port are shared which means we will need to experiment with the quickest method to send and receive data. Also, since there will be a lot of data sent from the first two boards, we will need to experiment with the best way to buffer the data before sending to prevent the loss of information.

G. Block 4: Raspberry Pi 4

The fourth Raspberry Pi board is dedicated to being the image processing node. The board will not have direct access to the videos the first and second Raspberry Pi records, so it will need to request the unprocessed video from the NAS board. The NAS will then send the video to the fourth board where it will process the image and then send the processed image back to be stored.

Image processing is a crucial step in the project because that is how we will determine if there is a pothole when the driver presses the button and how we will get size measurements of the pothole. In order to begin the process we had to research good algorithms and methods of edge detection, which is finding the edges of objects in a picture. The current method we are using to detect edges is the Canny Edge detection method [9] which determines if a pixel is an edge based on an algorithm. The algorithm checks the main pixel and its surrounding pixels to determine its magnitude and angle. It then uses its angle to check the pixel in the corresponding region to make sure it is in the correct magnitude threshold. If it is correct, the pixel will be marked white which indicates it is an edge.

After outlining all the edges in the picture, the picture will need to cleared up for any lines that are not an edge of the pothole. After the image has been cleaned up, the size will be determined based on the knowledge we know of the angle of the camera and the distance it is from the center point, which is eleven feet. These steps will be accomplished with algorithms.

Activity	October	November	December	January	February	March
Design Overall Project						
Design Two Camera Boards						
Design NAS						
Design Image Processing Board						
Build Camera Boards						
Build NAS						
Build Image Processing Board						
Build Database						
Take Picture, Accelerometer Reading, and GPS and send to NAS						
NAS Receive/Send Data						
Process Image and Extract Size/Depth						
NAS Send to Database						
Webpage Displaying Data						

Figure 7: Gantt chart showing current progress and future progress in the group.

Once the important data has been extracted by the algorithms, it will be sent back to the NAS to be stored. The data sent back will be the size, depth, the processed image, and the image of the original pothole. These images will be a frame taken from the video from the best image of the pothole.

We will need to have efficient codes and algorithms so the Raspberry Pi board will be able to handle all the processing that it will have to perform. Each press of the button requires two videos to be processed by the fourth board. To help with the processing we decided it was best to have a dedicated board to process the image. Though we have a dedicated board, we still need to be cautious of its processing power.

H. Block 5: Database and Webpage

The database stores the location, size, depth, and accelerometer reading of all potholes. The webpage grabs this data and plots the locations onto a map. This allows any user to now exactly where a pothole is and be able to prioritize which potholes to repair first.

III. Project Management

We currently have one board that has the camera interface, accelerometer, and GPS working. The board is able to take pictures and store them as bitmap files. The other board is the image processing board. We have been able to process the image to outline the pothole but we are still working on improving the processed image to make it more ideal to extract the size and depth. Our progress for MDR was greatly affected due to having to go back to designing stage of our project. The parts were ordered later due to this fact and thus our ability to have each subsystem working was not possible with the given amount of time. Due to the group falling behind where we wanted to be by MDR, we plan to accomplish a decent amount over winter break. Daniel and Muhammad will improve the image processing to a point that a size and depth can be extracted. Daniel will set up the NAS and look into being able to send and receive information between boards. The integration of the subsystems is a crucial part of the project that needs to be done before CDR. We want to be able to send a picture form one board to the NAS, which then will send it to the image processing board. The image processing board will extract the size and depth and send back the information to the NAS. The NAS will then send the data to the database. Once this is in place we can start to add the features and specifications that we wanted like integrating the accelerometer and GPS reading.

Each of our group members offers skills or expertise that will contribute to the project. Michael has experience doing group work involving both circuit design and computer programming. His experience with Java, C, and some Python will be useful when programming the Raspberry Pi. His circuitry skills will be invaluable when interfacing with the Raspberry Pi's hardware components. Various elements that will require hardware design in this project include the button that the driver activates when encountering a pothole and the accelerometers that will collect data regarding potholes. Michael is very interested in many disciplines of electrical engineering. His coursework focuses primarily on microwave engineering and signal processing. His knowledge of signal processing will be utilized when analyzing the data collected by the accelerometers. While his primary interests lie in electrical engineering, he also enjoys exploring interdisciplinary design problems. This curiosity will help when designing an enclosure to contain the finished project. Michael will also need to find a portion of a car frame to mount the accelerometers onto in order to collect pothole information.

William has experience building and verifying physical circuits in a group environment. He helped construct a switching voltage regulator using an efficient combination of several subsystems. This project improved his ability to verify that subsystems work together as planned. With regards to software William has learned multiple programming languages

through coursework and his independent project of building an Android app. William will lead the synchronization of the GPS, camera and accelerometer subsystems. He will design an algorithm to assign the most appropriate GPS location to data collected from the cameras and accelerometers. His experience synchronizing the physical subsystems of a switching voltage regulator and familiarity with multiple programming languages will be valuable in the synchronization phase. William is also knowledgeable of networks and communications from engineering course work, he will provide assistance with the NAS and WIFI components if Daniel or Muhammad request it.

Daniel has experience in working in groups on both coding projects and database projects. He has experience in coding in C, C++, Java, and Visual Basic. Daniel was able to put his programming knowledge to the test during his internship at Verizon. Here he combined his programming knowledge with setting up and creating databases using both Visual Basic and MySQL. These skills will come in handy when programming the boards and working on image processing. Daniel also has experience in networking through courses he has taken in college. This skill will be help when handling data from all three Raspberry Pi's along with sending data to the database. Though he might need to learn Python, it shouldn't be hard has experience learning new languages.

Muhammad has extensive experience in web development and working with databases in MySQL. He is able to run efficient queries through the database to get desired content and also able to present the data onto a webpage. He has created many websites and added content to existing ones through his internship at the DOT. Muhammad also has experience working with MySQL databases with C/C++. These skills will come in handy when the data from the NAS will be sent to be stored into the database and then displayed onto a map on a webpage. Muhammad has had experience coding in C/C++ throughout his years in college, which will help with the overall software implementation of the Raspberry Pis.

There are two CSE's and two EE's in our group therefore each person has one other to ask for help in terms of their area. Daniel and Muhammad work together on the software end of the boards, the majority of which is the image processing. Mike and Bill help each other in implementing all the devices like accelerometer and GPS and wiring them up. We are able to help each other out too because most devices have a software end to them and each of the boards have a hardware part.

Our team has been efficient with communication. With the convenience of text messaging we are able to communicate at all times. Whenever files or documents are involved, email and Google Docs are used.

IV. Conclusion

We have currently completed the majority of our image processing and camera board sub systems. We had multiple iterations of designs and had to make major changes after PDR. Due to this delay we had a late start to the actual building of sub systems. Our next step is to build the NAS and then integrate the subsystems. We will be able to send data between boards and to the database. The accelerometer will also be set up and we will have the specifications of where it will be placed on the car. The GPS and time stamping algorithms will also be implemented.

Our greatest difficulty will be the integration of subsystems, the network in particular. Sending and receiving data between boards gets difficult, especially with images and videos. We will have to set up and follow a TCP protocol and learn to send packets from one board to another.

Acknowledgment

We would first off like to thank our faculty advisor Professor Maciej Ciesielski for making time in his busy schedule to help us through the difficult process of designing and implementing our project. We would also like to thank our faculty evaluators Professor Bill Leonard and Professor Sandip Kundu. Last but not least we would like to thank Professor C.V. Hollot for his patience and helpful advice.

References

[1] "Adafruit Triple-Axis Accelerometer - Default Title." *Adafruit Triple-Axis Accelerometer*. Web. 10 Oct. 2015. <https://shop.pimoroni.com/products/adafruit-triple-axis-accelerometer>.

[2] "Adafruit Ultimate GPS HAT for Raspberry Pi A /B /Pi 2 - Mini Kit." *Adafruit Industries Blog RSS*. Web. 10 Oct. 2015. <https://www.adafruit.com/products/2324>.

[3] A. detection, 'Accelerometer data smoothing filtering pothole detection', *Electronics.stackexchange.com*, 2015. [Online]. Available: http://electronics.stackexchange.com/questions/56238/accelerometer-data-smoothing-filtering-pothole-detection. [Accessed: 09- Dec- 2015].

[4] "Camera Module - Raspberry Pi." *Raspberry Pi Camera Module Comments*. Web. 10 Oct. 2015. <https://www.raspberrypi.org/products/camera-module/>.

[5] E. Buza, S. Omanovic and A. Huseinovic, 'Pothole Detection with Image Processing and Spectral Clustering', University of Sarajevo, 2005.

[6] "HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi." *Cases for Your Raspberry Pi*. Web. 02 Nov. 2015. <http://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>.

[7] "Raspberry Pi - Teach, Learn, and Make with Raspberry Pi." *Raspberry Pi Home Comments*. Web. 20 Oct. 2016. <https://www.raspberrypi.org/>.

[8] Seattle.gov, 'SDOT - Seattle Pothole Information', 2015. [Online]. Available: http://www.seattle.gov/transportation/potholes/. [Accessed: 09- Dec- 2015].

[9] *Wikipedia*. Wikimedia Foundation. Web. 22 Oct. 2015. <https://en.wikipedia.org/wiki/Canny_edge_detector>.

[10] WUSA9, 'AAA: pothole damage costs drivers $6.4 billion a year', 2015. [Online].Available:http://www.wusa9.com/story/news/nation/2014/02/24/potholes-damage-cost-us/5773501/. [Accessed: 09- Dec- 2015].