Project and Mini-Project Proposal

Brett Baumert, Curtis Johnson, Michael Kelly
CSCE 488/489
Computer Science and Engineering
University of Nebraska-Lincoln
mkelly@cse.unl.edu
http://www.obdme.com

October 22, 2010

1 Introduction

From day one, our group knew that we wanted to do a project in which we came up with our own idea and project focus. We wanted to come up with an idea and focus that really interested all of us, and one that we could truly develop into a meaningful project. The ideas for our project stemmed from many different things. There were many motivating factors involved in the process in which we finally decided on a specific idea and focus for our senior project.

Our group initially formed because all three of us were somewhat interested in On Board Diagnostics (OBD) and what we could potentially do with that technology. All of us had a general idea of what OBD-II interface was and some of us had even experienced retrieving data from our vehicles via the OBD-II port. We were rather unsure as to what our exact project specifications would be, but from the beginning we knew that we were going to focus our project around the OBD-II technology.

As we researched OBD-II more, we quickly became aware that our project could go in many different directions, and there were many possibilities as to what we could do with this data. OBD-II data is standard on every vehicle post 1996. All of us have vehicles that are model year post 1996, so there were no concerns there. The idea of writing applications for mobile devices also entered our minds as a result of the discussion concerning the writing of mobile applications for the bus system. For the simple reason that all three of our group members have Android phones, we quickly became attuned to the idea of writing a mobile application for Android that would somehow connect with the OBD-II port of a vehicle and do many things with the data retrieved.

After some research, we realized that Bluetooth would be the ideal technology to transmit data between the OBD-II device and the Android powered mobile phone. Thus, our project focus was established.

Our project would have both a hardware aspect as well as a software aspect. The Bluetooth transmitter that interfaced with the OBD-II port would be the hardware aspect of our project,

and the Android mobile application as well a server side application would make up the software portion of our project. A more detailed description is given later in this report. Expanding our project to the scope of using it on other modes of transportation is not our main focus, but it could possibly be done if time allows.

Overall, our motivations came from the desire to incorporate OBD-II data and an Android application together. All of us are very interested in these technologies and feel that a very successful project can come from them.

2 Projected Functionality

When considering the goals for our mini project, we wanted to accomplish something that would be used directly in our overall design project. While we currently have a working OBD-II to Bluetooth device for testing purposes, we believe that we can implement the same functionality in a cheaper and more compact system. In order to test the functionality of our custom hardware solution, we decided that a simplified version of our target mobile phone application would also require development.

For the hardware device that will connect to the cars OBD-II port, we have two main routes that we could potentially pursue. We have the option to use a micro-controller with a pre-assembled and attached Bluetooth device or we could build our own controller and interface with a Bluetooth device directly. Each option has benefits and disadvantages in their own right, discussed in detail later in this report. Regardless of the route chosen, each potential hardware device will have the same functionality from the users perspective.

The hardware device will simply plug into the cars OBD-II port. Because power is supplied through the port, no other external connections are necessary. We also believe that security is not of great importance considering the data made available is read-only and of no use to anyone but the owner of the car. Therefore, when pairing the device with the mobile phone, the industry standard pin for embedded devices of either 0000 or 1234 will be used. Once the device has been paired with the mobile phone, a LED on the device will illuminate to indicate that the pairing was successful. For a typical driver, the current manifold air pressure is of no interest compared to something such as fuel economy. However, a car mechanic would be concerned with both the manifold air pressure and fuel economy as well as many other values. Due to this separation of concerns, the decision was made to develop an Android application with different display modes, each having varying levels of verbosity depending on the end user of the system. For the mini-project, the development focus of the application will be invisible to all end users and primarily concentrated on interfacing with the hardware device.

While mobile phones provide a valuable interface for viewing data from the hardware device, it is difficult to fully represent that data on a small display. Considering this obstacle, it was decided that a web interface would provide the perfect compliment to the mobile phone application. This feature introduces additional complexity in the design of our application but, provides a nearly limitless potential with manipulation and computation of the stored data.

As part of our final project design, the mobile phone application will read sensor data from the OBD-II hardware and additionally attach sensor data from the mobile phone such as GPS location and compass heading. This data will be continuously uploaded to a remote web service and stored in a database. With this stored data we can then present a wide variety of information

to the user in a more useful way than we could using a mobile phone application. Ultimately, our target system will give both the user and the mechanic relevant and useful diagnostic information. From a mechanics perspective, this history of data has a great potential to rule out or bring into consideration possible problems with a cars systems. From a end users perspective this will help them monitor their driving habits or the driving habits of family members.

3 Proposed Approaches

3.1 Hardware Interface

The first aspect of our project (and most important) will be the feature of interfacing with the standard OBD-II port that has been equipped in vehicles since 1996. We have a couple approaches to this problem, which have trade-offs of difficulty compared to price. Currently, an integrated circuit exists (ELM327) that enables communication to the OBD-II protocol via serial (RS-232) protocol. This circuit will be used in either of our approaches, as it is relatively inexpensive and performs most of the heavy lifting in terms of sending and receive messages via the OBD-II protocol. For the core functionality, we would like to send and receive messages to the ELM327 chip via Bluetooth. We have constructed two ways to approach this goal.

The first approach would be to purchase a microcontroller with a pre-installed Bluetooth module, and connect the ELM327 chip to the microcontroller. This way, any messages to and from the ELM327 chip would have to pass through the microcontroller first. This option would be rather trivial to implement, as we are all familiar with microcontrollers. However, it is not the most cost-effective, and might prove to be rather bulky.

The second approach would be to purchase a Bluetooth module separately, and connect it directly to the ELM327 chip. This approach seems to be the most popular, as it is more cost-effective (no need to purchase a microcontroller), and would be much smaller than the previous approach. However, this approach would require more work to implement.

In any approach, the hardware interface should provide a stable way of sending and receiving data to/from the engine computer in a vehicle.

3.2 Android Application

The next fundamental design piece of our project will be the Android application, which will have several main functions. First, it will act as the main display of real-time data from a vehicle. This will include displaying instantaneous and average statistics from the vehicle, such as intake air temperature or engine coolant temperature. The application should be versatile enough to allow the user to choose specific statistics to display.

The second key feature of the Android application will be the functionality to display specific driving habits to the user such as acceleration, deceleration, and fuel efficiency. These types of information will be most desired by drivers who are not worried about statistical values, but about the efficiency of their driving and the effects it may have.

Finally, the phone will act as the primary data ”messenger”, meaning the application will need to retrieve the data from the OBD-II port, and will also need to determine which data needs to be

logged. Once data is too be logged, the Android application will make a call to a data service via the HTTP protocol to log the data for future use.

3.3 Data Storage

Once the interface with the OBD-II port is successful, we will be able to obtain massive amounts of information pertaining to the car. This data will be valuable, whether the user is interested in average/instantaneous statistics, or data that is mixed with other data to solve engine troubles. But after we are presented with this data, we will need to keep a log of the data to be able to present the data to the user and possibly analyze it at a later time.

For a small scale project such as this, we will not have to require a large storage overhead, such as a clustered storage solution. However, we will need a dependable solution for storing this data, as the data may become vital towards future statistics. The sensible solution would be to store this data in a database.

Any outside applications will not be able to interface with the database directly, so another 'gateway' will need to sit in front of the database to accept and retrieve queries for clients. For this gateway, we would like to use one or several web services, depending upon our usage needs. Using these services, we can also handle authorization and other security-related issues, which will be important in guarding this data.

3.4 User Interaction

Finally, we would like to be able to present any data retrieved from a specific vehicle to a user through a web interface. This way, a user would have a intuitive way to view statistics about their vehicle and driving habits. Potentially, this interface might be used the same by mechanics to diagnose vehicular problems.

To present these statistics to the user, we would like to use web standards to provide compatibility between many devices from desktop computers to mobile devices. For this reason, we will develop a web application using HTML5 to allow the user to interact with their statistics.

3.5 Server Architecture

For the backend aspect of the entire system, we will require many dependable yet cost-effective solutions to keep the system responsive as well as easy to use. It is easy to see that the database could be possibly the most vital part of the back-end. Because of this, we have chosen to use an industry standard database system. MySQL fulfills many of our requirements, since it is cost effective (free), and also very dependable.

A sturdy web service structure and web application is also as important as the database for the full operation of this system. To implement these aspects of the project, we have chosen to use Java for our web framework, since it is also cost effective (free), dependable, and very familiar to the developers.

Finally, we will be running an Ubuntu server, hosting the MySQL database as well as the Java/JSP web application.

4 Risk Analysis

There are many potential risks that are presented with a project of this size and scope. The risks discussed below only represent the immediately identifiable potential risks. What is also of concern is the risks that are not identified and discussed in this report. Regardless of all potential risks, we are confident that we will be able to overcome them and complete the project on time and in full.

With a team of three senior computer engineering undergraduate students, schedules are very tight. Each team member has full course schedule with their own respective set of homework assignments, projects, and other obligations. Each team member is also employed in a part time job, further decreasing our available time. While our senior design project holds a very high importance, it can not dominate our academic or personal schedules. Navigating around these issues will be difficult without excellent time management. We have created a shared team calendar to facilitate collaboration, scheduling work, and to find common free time.

Due to the scope and complexity of the project, there are many hardware and software obstacles that must be overcome. The team is not specifically familiar with most of the commercial hardware and software that will be used in the project. This unfamiliarity will require time for training and familiarization. We can not accurately gauge how long this learning will take. Our best option in this situation is to divide out the technologies between group members and designate each team member as the expert on their assigned technologies. By doing this we can decrease the amount of learning time and we will be able to collaborate and assist one another when working with these technologies.

Our group has no way to anticipate the outcome or limitations of both hardware and software components used in the project. However, from our combined experience in both the academic and corporate environments, we have learned to expect these potential problems in advance. We are confident that what we are trying to accomplish using hardware is feasible given that commercial products with the same capabilities are already available. We also free that utilizing widely used open source software in our project design makes a great deal of technical resources available to us.

Despite the potential risks when considering the hardware and software in our project, there is also a significant amount of potential physical risks involved as well. The sensor data we collect and our mobile application interface is dependent on real sensor values, testing these things in a moving automobile introduces potential safety risks, especially when the tester and driver are the same person. We need to place a significant importance on driving safety because, not only are we putting ourselves at risk, but other individuals as well. Testing our device and software with OBD-II error codes means, we will be causing intentional mechanical problems with a car. We will need to ensure that these intentional problems will not permanently damage the engine and that it will not place us or others at risk when driving the vehicle.

Considering all of the potential risks above and the unforeseeable risks not mentioned, we are confident that acknowledging them in advanced will allow us to compensate for them in the future. Realizing these risks now will also allow us to correctly manage our time in the future so that we meet all of our projected goals.

5 Mini Project

The idea or concept in which our mini-project will implement has been up for speculation for a little while, but we finally have something that we feel will fit well for this. Since our project will consist of both a hardware aspect as well as a software aspect, we will incorporate both aspects into the mini-project.

The main focus of the mini-project will be the hardware aspect of our project. As previously stated, we will be developing an interface to the OBD-II port to communicate the data from the OBD-II port via Bluetooth to an Android enabled phone. The bulk of our mini-project will be the construction of this transmitter.

There are Bluetooth OBD-II dongles for sale on the Internet that accomplish exactly what we want to do, but we feel that a good aspect of our project will be to make our own. We have purchased a Bluetooth OBD-II dongle, but this will be used solely for the testing of our software aspect while we develop our own Bluetooth OBD-II dongle.

There are two main parts in this transmitter. First is the ELM327 OBD to RS232 interpreter. This chip will receive data from the OBD-II port and convert it to the RS232 (serial) format. The second component is the Bluetooth transmitter. We will need use the Bluetooth chip to transmit the serial data received from the ELM237 chip. There are a few different ways of going about creating this Bluetooth OBD-II transmitter. There are many pros and cons to each approach. Cost and size are two huge factors in this decision.

A more expensive, simple, approach would be to buy an Arduino platform that has a Bluetooth module built in. Arduino is a common microprocessor that is useful in many applications. With this Arduino and Bluetooth combination, all we would have to do is connect some input/output pins from the Arduino to the ELM237 chip and our Bluetooth OBD-II transmitter would basically be complete. The Arduino microprocessor would be easy to interface with. This approach is much more expensive, but it would allow for us to focus more on other aspects of our project.

The approach we will most likely use to build our Bluetooth OBD-II transmitter is much smaller, cheaper, but more complicated. We will purchase the ELM237 chip described earlier, as well as a Bluetooth transmitter. This approach lacks the Arduino microprocessor. We will do all of the interfacing of the ELM237 chip and the Bluetooth transmitter. This approach will be a little more challenging, but will create a more compact and cheaper solution.

Along with the hardware aspect of this mini-project, we will also implement a portion of the software aspect. Our goal is to have a basic Android application developed to communicate and receive data from the wireless Bluetooth transmitter attached to the OBD-II port. This application will basically display the data being received in a simple form.

6 Schedule

We have identified the following action items that will need to be included in our project and mini project schedule. Since this is a living document, we expect this list to grow and change as we progress through our objectives in the project.

6.1 Action Items

Advisor Meetings Meet weekly with Dr. Seth to evaluate and discuss our progress on the project.

Team Meetings Meet weekly as a group to work and discuss the project and each member's independent progress.

Mini Project Focus on mini-project this semester, and develop further if time permits.

Order Components Order component parts when necessary through the CSCE department.

Bluetooth Device Order the Bluetooth device to communication with the mobile phone

ELM 327 Order the ELM327 Chip to interface with the OBD-II port

Project Our final project design

Server Setup We will need a working web server, application server, database server, and subversion server

Android Application We will need to consider development efforts and make related milestones

Web Application We will need to consider development efforts and make related milestones

Imagine Cup We will need to enter the embedded development division.

Round 1 Initial Competition

Start Date July 9, 2010

End Date January 9, 2011

Round 2 Semi-finals

Start Date February 15, 2011

End Date May 5, 2011

Round 3 Worldwide Finals

Start Date July 2011

End Date July 2011

6.2 Monthly Target Dates

October 31, 2010 Have parts ordered and begin development of the hardware interface.

November 30, 2010 Have a basic Android application developed. This application will communicate with the developed hardware.

December 31, 2010 Development of the mini-project should be coming to an end, including implementation of the hardware OBD-II device, as well as beginnings of the Android application.

January 31, 2011 By this time, a majority of the Android application should be developed, and the necessary data/web interfaces should be realized.

February 28, 2011 A database structure should be implemented, as well as web services to allow outside devices to interact with information in the database.

March 31, 2011 Server-side implementation of learning-algorithms, including algorithms to determine problem factors/causes, as well as driving habit analysis.

April 30, 2011 A web application should be developed and implemented to allow users to view/interact with data about their vehicles that has been retrieved via the OBD-II interface.