Auto Lynk OBD-II Scanner

Chris Frey
May 3, 2013
ECET - CPET 491 Senior Design Project Phase II
Project Advisor and Instructor: Professor Paul I. Lin

Summary

- Goals
- Motivation
- Introduction
- Problem Statement / Solution
- Project Research
- Overall Project Design
- Hardware Interface
- Software Design
- Testing and Integration
- Project Management
- Conclusion

Goals

- On-Board Diagnostic-II (OBD-II) scanner.
  - Android-based
  - Connection to an OBD-II Bluetooth adapter
  - Monitor various vehicle subsystems.
  - Users able to identify problems with their vehicles.

3

Motivation

- Conquer my fear of programming
- Interest in smartphones
- Interest in automotive systems

- Combining these = success?

4

Introduction

- History of On-Board Diagnostics
  - 1950's - Problems diagnosed by hand
  - 1960's - Vehicles became more complex
    - Humans being removed from the loop
  - 1980's - Emerging emission standards
    - Malfunction Indicator Lamp (MIL) became required
  - 1987 - California required OBD-I

5

OBD-II

- Vehicles produced after 1996
- Monitors various vehicle subsystems (Body, Powertrain, Chassis, Network)
  - Values such as:
    - Engine Load, Oxygen Sensor Voltage, MPH, RPM
- Diagnostic Trouble Code (DTC)
  - Stored when MIL illuminates

6

Problem / Solution

- How do we read OBD-II data?
- Where do we access it?

- Answer:
  - Adapter
  - Application that allows OBD-II data to be read on a smartphone.
  - Reset DTCs when vehicle problem has been solved

7

Project Research

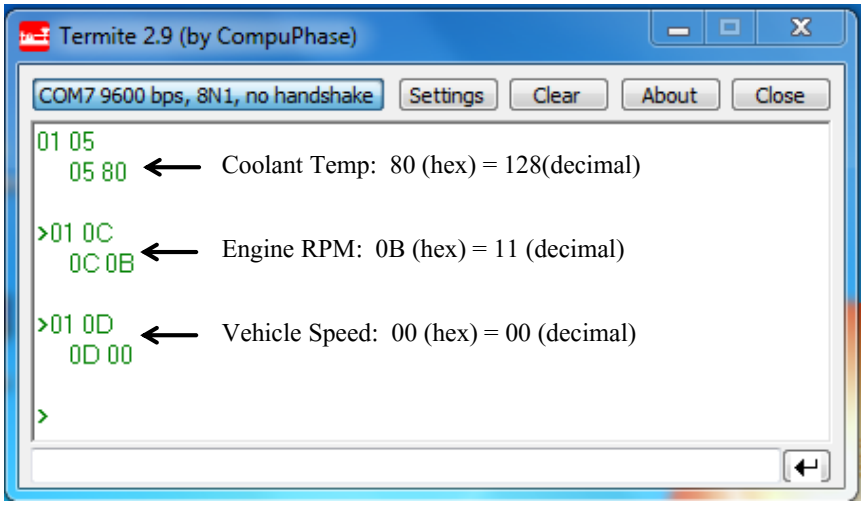
- OBD-II Specifications
  - Standardized hardware interface
  - Presented in C/C++ programming language
  - Parameter ID (PID) message is sent to the vehicles Engine Control Unit (ECU)
  - Value returned in hexadecimal format

8

OBD-II Specifications (cont.)

PID	Description	Min Value	Max Value	Units	Formula
04	Engine Load	0	100	%	A * 100 / 255
05	Engine Coolant Temperature	-40	215	°F/°C	A - 40
06	Fuel Trim Bank 1 Sensor 1	-100	99.22	%	(A-128) * 100/128
07	Fuel Trim Bank 1 Sensor 2	-100	99.22	%	(A-128) * 100/128
0B	Intake Manifold Pressure	0	255	kPa	A
0C	RPM	0	16,383.75	Rpm	(A*256) / 4
0D	Speed	0	255	mph	A
0E	Timing Advance	-64	63.5	°	A/2 - 64
0F	Intake Air Temperature	-40	215	°F/°C	A – 40
11	Throttle Position	0	100	%	A*100/255
14	Oxygen Sensor Bank 1 Sensor 1	0	1.275	Volts	A/200
15	Oxygen Sensor Bank 1 Sensor 2	0	1.275	Volts	A/200
AT RV	Voltage	0	15	Volts	A

Initial Testing (Termite)



10

Initial Testing (cont.)

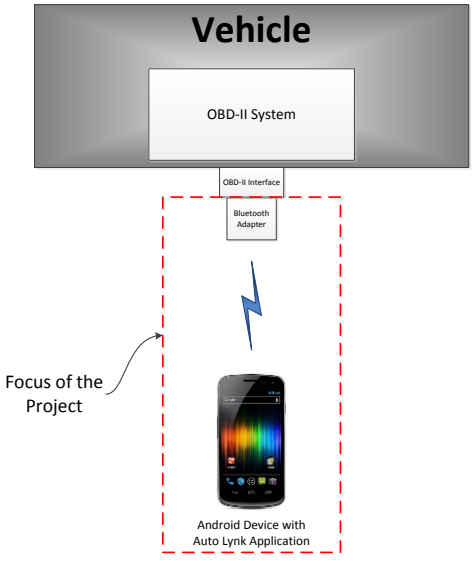
Engine Coolant Temp
A – 40 = (128 - 40) = 88°C = 190°F

Vehicle Speed
A = 0 km/h = 0 MPH

Engine RPM
(A * 256) / 4 = ((11) * 256) / 4 = 704 RPM

11

Overall Project Design



12

Hardware Interface

- Soliport ELM 327 OBDII Bluetooth Adapter
  - Allows communication between OBD-II port and smartphone via Bluetooth
  - Bluetooth Serial Port Profile
    - Emulates RS-232
  - Sends and receives ASCII values
  - Services one command at a time

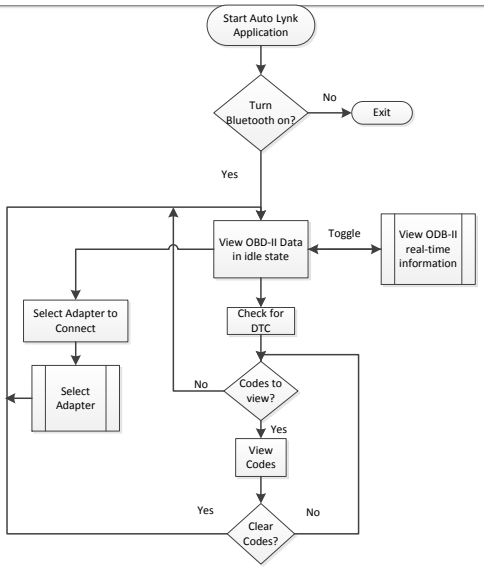
13

Auto Lynk Design

- Simple and easy to use
  - List format
  - Everything on one screen
- Review Requirements
  - Reads OBD-II data
  - Bluetooth Connectivity

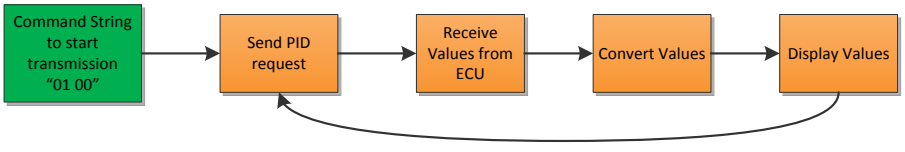
14

Auto Lynk Design



15

Auto Lynk Algorithm



- Establish constant data flow between ECU and Android device
- Uses Java Switch statement

16

Auto Lynk – The Code

- Eclipse IDE
- Android Software Development Kit (SDK)
- Android Virtual Device (AVD)

17

BluetoothChat Example

- Enables Bluetooth
- Scans for nearby Bluetooth devices
- Connects to a Bluetooth device
  - Creates socket to communicate between 2 devices
- Modified Universally Unique Identifier (UUID)

```
public ConnectThread(BluetoothDevice device, boolean secure) {
    mmDevice = device;
    BluetoothSocket tmp = null;
    mSocketType = secure ? "Secure" : "Insecure";

    // Modified to work with SPP Devices
    final UUID SPP_UUID = UUID
            .fromString("00001101-0000-1000-8000-00805F9B34FB");
```

18

Sending Requests to the ECU

- Start Transmission with "00 01" message
- getData() method

```
public void getData(int messagenumber) {

    final TextView TX = (TextView) findViewById(R.id.TXView2);

    switch (messagenumber) {

    case 1:
        sendMessage("01 0C" + '\r'); // get RPM
        TX.setText("01 0C");
        messagenumber++;
        break;

    case 2:
        sendMessage("01 0D" + '\r'); // get MPH
        TX.setText("01 0D");
        messagenumber++;
        break;
```

19

Parsing the Incoming Data

- Regular Expressions
- Switch Statement

```
if ((dataRecieved != null)
        && (dataRecieved
                .matches("\\s*[0-9A-Fa-f]{2} [0-9A-Fa-f]{2}\\s*\r?\n?"))) {

    dataRecieved = dataRecieved.trim();
    String[] bytes = dataRecieved.split(" ");

    if ((bytes[0] != null) && (bytes[1] != null)) {

        PID = Integer.parseInt(bytes[0].trim(), 16);
        value = Integer.parseInt(bytes[1].trim(), 16);
    }

    switch (PID) {

    case 15:// PID(0F): Intake Temperature

        value = value - 40; // Formula for Intake Temperature
        value = ((value * 9) / 5) + 32; // Convert from Celsius to Farenheit
        String displayIntakeTemp = String.valueOf(value);
        intakeTemperature.setText(displayIntakeTemp + " °F");
```

20

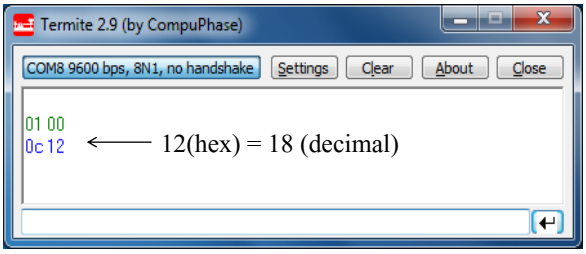
Resetting Trouble Codes

- Easiest to implement
- clearCodes() method

```
public void clearCodes() {

    final TextView TX = (TextView) findViewById(R.id.TXView2);

    if (mConnectedDeviceName != null) {

        sendMessage("04" + '\r'); // send Clear Trouble Codes Command
        TX.setText("Clear Codes");
        Toast.makeText(getApplicationContext(),
                "OBD Trouble Codes Cleared", Toast.LENGTH_SHORT).show();
```

21

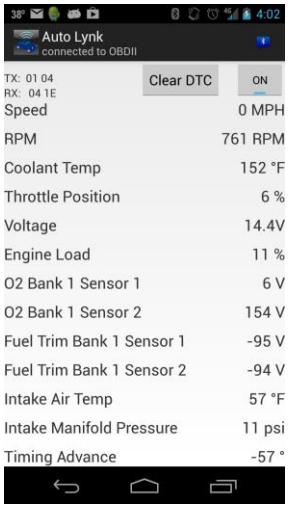
Testing and Integration



$(A * 256)/4 = ((18) * 256) / 4 = 1152$

22

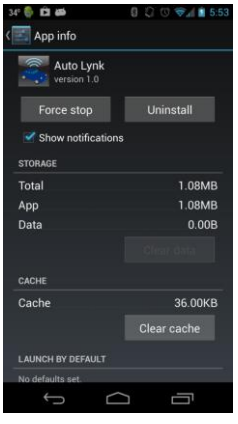
Testing and Integration (cont.)



23

Validation of Requirements

- Most requirements shown in demo



24

Validation of Requirements (cont.)

- Diagnostic Trouble Code Reader
  - Deadline approaching
  - Code more difficult than expected
    - Sheer number of codes
- Will be implemented in future

25

Bill of Materials

Description	Supplier	Price
Soliport ELM 327 OBD-II Bluetooth Adapter	www.amazon.com	$22.43
RocketFish Micro Bluetooth Adapter	Best Buy	$20.00
Window 7 PC	Myself	$0.00
Java SE Development Kit (JDK) 6	www.oracle.com	$0.00
Android Software Development Kit (SDK) Includes: Eclipse ADT plugin Android SDK tools Android Platform-tools	developer.android.com/sdk/index.html	$0.00
Samsung Galaxy Nexus smartphone w/ Android 4.2.2	Myself	$0.00
	Total	$42.43

26

Work Breakdown

Tasks	Estimated Hours of Completion	Actual Hours
System Design	30	40
Assembly Phase	58	77
System Testing	35	35
Final Report Development	15	32
Presentation Development	10	7
Total	**150**	**191**

27

Schedule

- System Design Completed – February 18, 2013
  - Research
- Assembly Phase Completed – April 3, 2013
  - Write Auto Lynk
- System Integration Completed – April 25, 2013
  - Testing
- Final Report Complete – May 2, 2013
- Presentation Complete – May 3, 2013

28

Lessons Learned

- Time Constraints are crucial
- Attention to detail
- Organizational skills
- OBD-II
- Programming fear conquered!!

29

Conclusion

- Provided research for OBD-II and Android
- Auto Lynk OBD-II Scanning system was successful, although not complete
- More features to be added in future
  - DTC reader
  - Others

30

Questions??

Demo