



UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

IF747 – REDES AUTOMOTIVAS

PROJETO – PARTE 1

PROFESSOR: Divanilson Campelo

MONITOR: Paulo Freitas

Rubem Moura

Thaysa Barros

Recife – 2016

PROPOSTA

Como parte do projeto de Redes Automotivas foi lançada a proposta de implementar um sistema de captura de dados de um veículo em tempo real, cujos dados deveriam constituir um banco de dados. Dentre as solicitações, é necessária a tradução desses dados para que em outra etapa do projeto, as informações coletadas sejam usadas em simulações e aplicações.

ETAPA 1 – TESTES EM CAMPO

Baseado nas solicitações do projeto, o primeiro passo foi adquirir um Scanner Automotivo OBD2, o ELM 327, que envia dados aos aplicativos via Bluetooth. Com o ELM 327 foram realizadas três coletas de dados em um Ford Fiesta 2014, cada uma com um aplicativo diferente, o Torque Free, Torque Pro e OBD Doctor, sendo os dois primeiros usados em celular e o último em um notebook. No entanto, as duas versões do Torque não forneciam dados no formato de frames, que foi estabelecido como objetivo do projeto, e quanto ao OBD Doctor, a aquisição de dados não foi iniciada, apesar de estarem conectados.

Diante dos resultados negativos, outro Scanner foi utilizado, Vgate Bluetooth Wireless, e o OBD Doctor no notebook comunicando-se via Wifi. Nessa configuração, foram obtidos os frames de quatro sensores de um Ford Ka 2015, os quais informam velocidade, rotação, temperatura de resfriamento do motor e temperatura do ar que entra para participar da combustão, essa quantidade de sensores não confere com o solicitado, 5 sensores, mas dentre os muitos sensores disponíveis, esses foram os únicos que geraram informação através desse aplicativo. Os scanners podem ser visualizados na Figura 1 e as interfaces de comunicação, na Figura 2 do Apêndice.

ETAPA 2 – INTERPRETAÇÃO DO DADOS

Assim, com um arquivo txt gerado pelo OBD Doctor, Figura 3, os dados foram transferidos para o Excel, Figura 4, onde as informações são facilmente separáveis por colunas, a fim de facilitar a visualização e manipulação do frame e a interpretação dele. Nessa figuras com trechos dos dados coletados, está indicada a coluna de direção, que significa o sentido de envio da mensagem: >> indica o envio de uma mensagem de requisição da ECU para o sensor e << indica a mensagem de resposta que parte do sensor e segue para a ECU.

Abaixo seguem exemplos de mensagem requisitada, Tabela 1, e mensagem de resposta, Tabela 2, separados de acordo com o significado de cada caractere.

MODO	PID
01	05

Tabela 1 – Mensagem-requisição

PID TYPE	BYTE						
	0	1	2	3	4	5	6
PID	ADDITIONAL DATA	SHOW CURRENT DATA	DATA	DATA	DATA	DATA	DATA
7E8	03	41	05	7F			
7E8	04	41	0C	0E	EF		
7E8	03	41	0D	00			
7E8	03	41	0F	54			

Tabela 2 – Frame de mensagem-resposta.

Acompanhando a tabela mensagem-requisição, o modo 01, o único que aparece neste teste, pede aos sensores que mostrem seus valores em tempo real, enquanto o PID revela o sensor ao qual a mensagem é destinada.

Seguindo para a tabela de mensagem-resposta, o PID TYPE é um identificador de uma ECU, o 7E8 é o identificador de resposta para a ECU #1 que corresponde ao módulo de controle do motor. As demais colunas formam o *payload*, são ao todo 8 bytes, mas os 6 da tabela serão suficientes para explicar os dados coletados. O byte 0 informa o byte máximo de dado que será enviado, nos exemplos tivemos dois casos, 03 e 04, observe que onde tem 03 o dado está preenchido até o byte 3 e o 04, até o byte 4. O byte 2 é um modo de consulta e 41 indica modo em tempo real, o byte 3 é a identificação do sensor e os seguintes são os que correspondem ao valor assumido pelos sensores. A identificação do sensor ou PID para os sensores utilizados no teste é: 05 para temperatura de resfriamento do motor, 0C para rotação, 0D para velocidade e 0F para temperatura do ar de admissão.

ETAPA 3 – TRADUÇÃO DOS DADOS

Para registrar a interpretação do *payload*, pois ficou acordado que os próximos passos do projeto seriam feitos com dado tratado, foi desenvolvido um programa em linguagem C que converte de hexadecimal para decimal os itens “DATA” do *payload*, Figura 5. Além disso, é necessário mais do que a simples conversão de base numérica, é preciso aplicar as fórmulas abaixo, segundo a referência [1]. As fórmulas 2 e 3 são, respectivamente, dos sensores 0C, 0D e a fórmula 1 é a mesma para os sensores 05, e 0F.

$$T = A - 40 \quad (1)$$

$$R = \frac{256A+B}{4} \quad (2)$$

$$V = A \quad (3)$$

Observe que o valor A corresponde ao decimal do primeiro byte de dado e B é o decimal do segundo byte de dado.

Em seguida, no Excel os dados não tratados foram filtrados de maneira que só as mensagens-resposta permanecessem, já que elas serão necessárias na etapa seguinte, a simulação, para então submetê-las a um programa de tratamento de dados.

No código, os dados de mensagem-resposta foram importados de um arquivo txt, iniciando a sequência de eventos que varre todo o banco de dados até que a conversão completa seja feita. Os eventos do programa consistem principalmente em importar um frame – Figura 5, a partir da linha 163, *main* –, converter de hexadecimal para decimal – Figura 5, funções da linha 6 a 70 – e salvar em um novo arquivo – Figura 5, linha 72, dentro da função *processPayload* –, repetindo sequencialmente esses passos até o fim do banco de dados.

APÊNDICE



Figura 1 – ELM 327 e Vgate.

Figura 2 – Interfaces para aquisição de dados.

TIMESTAMP	DIRECTION	DATA
2016-11-16 19:37:12	>>	0101
2016-11-16 19:37:13	<<	7E80641010007E100
2016-11-16 19:37:17	>>	0105
2016-11-16 19:37:17	<<	7E80341057F
2016-11-16 19:37:17	>>	010C
2016-11-16 19:37:17	<<	7E804410C0EFE
2016-11-16 19:37:17	>>	010D
2016-11-16 19:37:18	<<	7E803410D00
2016-11-16 19:37:18	>>	010F
2016-11-16 19:37:18	<<	7E803410F54
2016-11-16 19:37:18	>>	0105
2016-11-16 19:37:18	<<	7E80341057F

TIMESTAMP	DIRECTION	DATA
16/11/2016 19:37:12	>>	0101
16/11/2016 19:37:13	<<	7E80641010007E100
16/11/2016 19:37:17	>>	0105
16/11/2016 19:37:17	<<	7E80341057F
16/11/2016 19:37:17	>>	010C
16/11/2016 19:37:17	<<	7E804410C0EFE
16/11/2016 19:37:17	>>	010D
16/11/2016 19:37:18	<<	7E803410D00
16/11/2016 19:37:18	>>	010F
16/11/2016 19:37:18	<<	7E803410F54
16/11/2016 19:37:18	>>	0105
16/11/2016 19:37:18	<<	7E80341057F
16/11/2016 19:37:18	>>	010C
16/11/2016 19:37:19	<<	7E804410C0F65
16/11/2016 19:37:19	>>	010D
16/11/2016 19:37:19	<<	7E803410D00
16/11/2016 19:37:19	>>	010F
16/11/2016 19:37:19	<<	7E803410F54

Figura 3 – Arquivo txt gerado pelo OBD Doctor.

Figura 4 – Banco de dados no Excel.

```

6  int convertHexToInt(int hex_value, int indice_base){
7      int result = 0;
8      int base_hex = 16;
9
10     if(indice_base == 0){
11         base_hex = 1;
12     }
13     result = hex_value * base_hex;
14
15     return result;
16 }
17
18 int convertLetterToInt(char letter){
19     int result = 0;
20
21     switch(letter){
22         case 'A':
23             result = 10;
24             break;
25         case 'B':
26             result = 11;
27             break;
28         case 'C':
29             result = 12;
30             break;
31         case 'D':
32             result = 13;
33             break;
34         case 'E':
35             result = 14;
36             break;
37         case 'F':
38             result = 15;
39             break;
40     }
41     return result;
42 }

```

Figura 5 – Código de conversão do banco de dados.

```

45 int letterOrNotLetter(char letter){
46     int result;
47     if(letter == 'A' || letter == 'B' || letter == 'C' || letter == 'D' || letter == 'E' || letter == 'F'){
48         result = convertLetterToInt(letter);
49     }else{
50         result = letter - '0';
51     }
52
53     return result;
54 }
55
56 int captureOneByte(char byte0, char byte1){
57     //converte dois char para int
58     int value = 0;
59     int value2 = 0;
60     int valueInt = 0;
61
62     value = letterOrNotLetter(byte0);
63     value2 = letterOrNotLetter(byte1);
64
65     value = convertHexToInt(value, 1);
66     value2 = convertHexToInt(value2, 0);
67     valueInt = value + value2;
68
69     return valueInt;
70 }
71
72 int processPayload(char* byte, char* payload){
73
74     char sensorHb = payload[8];
75     char byte0[2];
76     byte0[0] = byte[0];
77     byte0[1] = byte[1];
78
79     char ch;
80     FILE *out;
81     out = fopen("result.txt", "a");
82
83     if(out == NULL){
84         printf("Erro, não foi possível abrir o arquivo\n");
85     }else{
86         // fseek(out, 7, SEEK_END);
87         int result = 0;
88         switch(sensorHb){
89             case '5':
90                 //Engine coolant temperature!
91                 //A-40
92                 printf(" Engine coolant temperature!\n");
93
94                 int coolantTemperature = 0;
95                 coolantTemperature = captureOneByte(byte0[0], byte0[1]) - 40;
96                 result = coolantTemperature;
97                 fprintf(out, "Coolant Temperature: %d\n", result);
98                 printf("\n Coolant Temperature: %d graus Celsius\n", coolantTemperature);
99                 break;

```

Figura 5 – Código de conversão do banco de dados.

```

100     case 'F':
101         //Intake air temperature
102         //A-40
103         printf(" Intake air temperature!\n");
104
105         int intakeAirTemperature = 0;
106         intakeAirTemperature = captureOneByte(byte0[0], byte0[1]) - 40;
107         result = intakeAirTemperature;
108         fprintf(out, "Intake Air Temperature: %d\n", result);
109         printf("\n Intake Air Temperature: %d graus Celsius\n", intakeAirTemperature);
110         break;
111     case 'C':
112         //Engine RPM
113         //((A*256)+B)/4
114         printf(" Engine RPM!\n");
115
116         char byte1[2];
117         byte1[0] = payload[11];
118         byte1[1] = payload[12];
119
120         int RPM = 0;
121         int A = 0;
122         int B = 0;
123
124         A = captureOneByte(byte0[0], byte0[1]);
125         B = captureOneByte(byte1[0], byte1[1]);
126
127         RPM = ((A*256)+B)/4;
128         result = RPM;
129         fprintf(out, "RPM: %d\n", result);
130         printf("\n RPM %d \n", RPM);
131
132
133         break;
134     case 'D':
135         //Vehicle speed
136         //A
137         printf(" Vehicle speed!\n");
138
139         int vehicleSpeed = 0;
140         vehicleSpeed = captureOneByte(byte0[0], byte0[1]);
141         result = vehicleSpeed;
142         fprintf(out, "Vehicle speed: %d\n", result);
143         printf("\n Vehicle speed: %d km/h\n", vehicleSpeed);
144         break;
145     }
146 }
147
148 fclose(out);
149 printf("\n\n-----\n\n");
150 return 0;
151 }

```

Figura 5 – Código de conversão do banco de dados.


```

153 int main()
154 {
155     char payload[] = "7E804410C1CEB";
156
157     char *ch;
158     int count = 0;
159     char string_frame[13];
160     FILE *frame;
161     frame = fopen("frames.txt", "r");
162
163     if(frame == NULL){
164         printf("Erro, nao foi possivel abrir o arquivo\n");
165     }
166     else{
167         while((ch=fgetc(frame)) != EOF){
168             while(ch != '\n'){
169                 string_frame[count] = ch;
170                 count++;
171                 ch=fgetc(frame);
172             }
173             string_frame[count] = '\0';
174             count = 0;
175             printf(" String_frame = %s \n\n", string_frame);
176             char byte0[2];
177             byte0[0] = string_frame[9];
178             byte0[1] = string_frame[10];
179
180             processPayload(byte0, string_frame);
181
182             *string_frame = NULL;
183         }
184     }
185
186     fclose(frame);
187     return 0;
188 }
189

```

Figura 5 – Código de conversão do banco de dados.

```

String_frame = 7E803410D1E
Vehicle speed!
Vehicle speed: 30 km/h

-----

String_frame = 7E803410F57
Intake air temperature!
Intake Air Temperature: 47 graus Celsius

-----

String_frame = 7E803410580
Engine coolant temperature!
Coolant Temperature: 88 graus Celsius

-----

String_frame = 7E804410C1BA7
Engine RPM!
RPM 1769

-----

Process returned 0 (0x0)   execution time : 27.089 s
Press any key to continue.

```

Figura 6 – Console com a conversão de dados.

```

Intake Air Temperature: 44
Coolant Temperature: 88
RPM: 999
Vehicle speed: 24
Intake Air Temperature: 45
Coolant Temperature: 88
RPM: 933
Vehicle speed: 22
Intake Air Temperature: 45
Coolant Temperature: 88
RPM: 912
Vehicle speed: 17
Intake Air Temperature: 45
Coolant Temperature: 87
RPM: 911
Vehicle speed: 12

```

Figura 7 – Novo arquivo com dados convertidos.

REFERÊNCIAS

[1] https://en.wikipedia.org/wiki/OBD-II_PIDs#Mode_01

[2] Material de Aula – IF747