STM32-based Vehicle Data Acquisition System for Internet-of-Vehicles

Yong Xie*, Xin Su†, Yifan He*, Xuhui Chen*, Gengliang Cai*, Baisheng Xu*, Wenjia Ye*
*Department of Computer and Information Engineering, Xiamen University of Technology, Xiamen, R.P.China
†Hunan Provincial Key Laboratory of Network Investigational Technology, Hunan Police Academy, Changsha, R.P.China
‡Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands.
Email: yongxie@xmut.edu.cn, suxin@hnu.edu.cn, y.he@tue.nl, xhchen@xmut.edu.cn,
1022994235@qq.com, 1251604436@qq.com, 1205244519@qq.com

***Abstract*—As the new era of the Internet of Things(IoT) is driving the evolution of conventional Vehicle Ad-hoc Networks into the Internet of Vehicles(IoV), vehicles are equipped with different kinds of sensor and become a sensing node themselves in IoV. Consequently, complexity of the automotive electronic system inside the vehicles are daily increasing. To guarantee the safe operation of vehicles, it is of great significance to acquire vehicle data in real-time to realize the on-line diagnosis, cyber-security attacking detection and et al. In this paper, we propose to realize a STM32-based data acquisition system(DAS), where the vehicle data transferred on CAN networks are acquired through the OBD2(On-Broad Diagnosis) interface. And then, the acquired vehicle data are parsed and analyzed preliminarily according to the OBD2 protocol, and then shown on a LED displayer. Through the implementation of a prototype system, the feasibility and effectiveness of the proposed design of DAS is verified.**

***Index Terms*—Internet of Vehicle, Data Acquisition System, On-board Diagnosis, Controller Area Network.**

I. Introduction

The new era of the Internet of Things is driving the evolution of conventional Vehicle Ad-hoc Networks into the Internet of Vehicles (IoV). With the rapid development of information and communication technologies, IoV promises huge commercial interest and research value, thereby attracting a large number of automotive companies, OEMs, IT providers and researchers [1], [2]. In a report from Business Insider, it is estimated that by 2020 75% of cars will be built with the necessary hardware to connect to the Internet [3]. Next-generation vehicles will not only precept the environment with their own sensors, but also communicate with other vehicles and surrounding infrastructures for vehicle safety and transportation efficiency.

To make this vision come true, modern automotive electronic systems have become more complex than ever, in terms of both electronic functionality and architecture. From the functional perspective, there is a wide range of emerging applications including autonomous functions and Advanced Driver Assistance Systems (ADAS), such as adaptive cruise control and lane keeping assist. To fulfill these applications, various software programs are implemented to play important roles in sensing, signal processing, control, decision making, etc. From year 2000 to 2010, embedded software increased from 2% to 13% of a vehicles total value, and the number of lines of code increased from one million to more than ten million [4]. From the architectural perspective, the number of Electronic Control Units (ECUs) in a standard car has gone from 20 to over 100 in the past decade [4], [5]. But the above mentioned developing trend poses great challenges to the safety guarantee of the vehicles, it is of great importance to acquire the real-time data inside the vehicles to implement the on-line diagnosis, cyber-security attacking detection, driver behavior analysis [6], [7], [8] and etc. However, vehicles are mass-produced consumer products, it is not cost-efficient to implement the above functions with ECUs(Electronic Control Unit). Therefore, as shows in Figure 1, it is a good choice to take advantage of the 4G/IPv6-based edge computing paradigm to the IoV, as it can meet the requirements of low latency, mobility, security, QoS guarantee and etc [9]. In this paper, we focus on the real-time data acquisition of vehicle data, and the data analysis in edge cloud server is remained to be presented in our future work. The main contributions are as follows: (1) we implement a STM32-based data acquisition system(DAS), where the the vehicle data transferred on CAN networks are acquired through the OBD2(On-Broad Diagnosis) interface; (2) vehicle data are parsed and analyzed preliminarily according to its data contents, and then shown on LED displayer. In our future work, the preliminarily analyzed data will be transferred to the edge cloud server through wireless IPv6-based WIFI for further analysis.

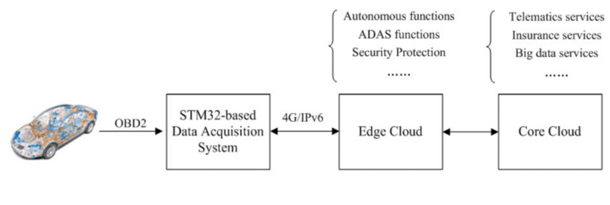


Fig. 1. 4G/IPv6-based Mobile Edge Computing Paradigm for Internet-of-Vehicles

II. System Architecture and Functions

A. System Architecture

As shown in Figure 2, this is the hardware architecture of the implemented DAS, where data transferred on in-vehicle CAN

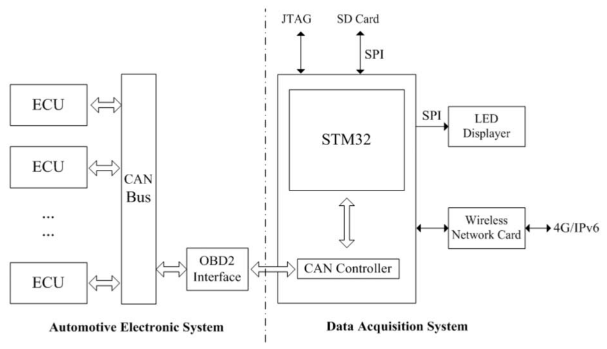


Fig. 2. Hardware Architecture of STM32-based Data Acquisition System

networks are acquired through the OBD2 interface. By analyzing the obtained data according to the OBD2 protocol(ISO 15765), we can get the vehicle status in real-time, such as the vehicle speed, engine speed and etc. On the one hand, the vehicle data can be demonstrated on a LED displayer. On the other hand, the vehicle data will be transferred to the edge cloud server for further analysis, such as the on-line diagnosis, driving behavior analysis, security intrusion detection and so on.

B. System Components

1) STM32F4: We use STM32F4 as the microcontroller of the DAS, STM32 F4-series is the first group of STM32 microcontrollers based on the ARM Cortex-M4F core. Common peripherals included in STM32F4 are USB 2.0, CAN 2.0B, SPI, I2S, UART, SDIO, ADCs/DACs, GPIOs, and etc. Please refer to [10] for more details about STM32F4.

2) OBD2: OBD2 interface gives the vehicle owner or repair technician access to the status of the various vehicle subsystems [11]. OBD2 standard specifies the type of diagnostic connector and its pinout, the electrical signalling protocols available, and the messaging format. There are five signaling protocols that are permitted with the OBD2 interface, and we implemented ISO 15765 in DAS as ISO 15765 standard is widely utilized among the on sale vehicles produced by USA and Europe automobile manufacturers [12].

3) LCD Displayer: We use red 2.4 inch Touch 320x240 SPI LCD module TJCTM24024-SPI as the LCD displayer, this is a ILI9341 driven LCD display, with XPT2046 chip for touch device, and it also includes a SD card holder.

III. System Implementation

In this paper, we implemented the ISO 15765-based OBD2 protocol to acquire the vehicle data. ISO 15765 protocol conforms to the classical ISO/OSI reference model, which allows it to be integrated with other underlying communication protocols. We integrate IS0 11898-based CAN protocol into the network layer and data link layer of ISO 15765 protocol, and based on it the vehicle data sent on CAN can be acquired successfully.

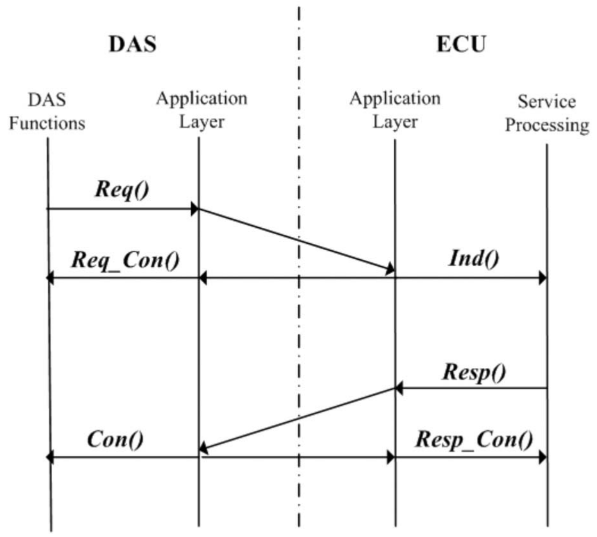


Fig. 3. Communication between Data Acquisition System and ECUs

In the application layer of ISO 15765, 6 primitives are defined to realize the communication between OBD2 accessing clients(DAS) and OBD2 servers(ECU). Among these primitives, $Req()$ represents the service request sent from DAS, and the corresponding function is $XXX - Req()$; $Req - Con()$ represents the acknowledge information request sent from DAS to confirm if the service request is received by ECUs, and the corresponding function is $Send - dReqCon()$; $Ind()$ represents the receiving of data request in ECUs, and the corresponding function is $SendInd()$; $Resp()$ represents the request response of ECUs and the requested data will be sent back to DAS, the corresponding function is $XXX - Resp()$; $Resp - Con()$ represents the data receiving confirmation of the DAS, and the corresponding function is $SendRespCon()$. The detailed communication process between DAS and ECU is shown in Figure 3.

Based on the above described application primitives, the DAS and ECU can communicate through the request/respond approach. When DAS sends a request service, this service will be sent to the ECU through the OBD2 interface. After the analysis of the request service, the corresponding ECU will send back the requested information through the OBD2 interface. There are 9 modes of operation are defined for the OBD2, for example ”Mode: $01” is used to identify what powertrain information is available to the accessing clients [11]. And based on it, different kinds of vehicle data can be acquired to the DAS. Each operation mode is followed by a PID(Parameter IDs) with two bits, which defines the data that can be acquired in this mode. Table 1 shows some common PIDs.

$$\begin{aligned} Req &: \ 08\ FBE0\ 02\ 01\ 0D\ 00\ 00\ 00\ 00 \\ Ans &: \ 08\ FD00\ 07\ 41\ 0D\ 4F\ 00\ 00\ 00 \end{aligned}$$

The above two lines describes an example of vehicle speed

TABLE I
SOME COMMON OBD2 PIDS

Description	SID	PID	Position	Unit	Calculation Method
engine load	01	04	3	%	$y = x * 100/255$
vehicle speed	01	$0D$	3	km/h	$y = x$
engine oil temperature	01	$5C$	3	°C	$y = x - 40$

acquisition, where the Req is the data request frame sent by DAS, and Ans is the answering frame from the ECU. In each CAN frame, the first 3 bytes are about the frame's information, such as frame ID, CRC and et al; and the left 8 bytes are the frame's payload. As shown in the Ans frame, the third byte of the payload(in red color) is 0X4F. Thus, according to the given calculation method, the obtained vehicle speed is 79 km/h. Next, we give the main implementation codes of the DAS as follows:

```
u8 ObdScan(){
//u16  StdID=0xFBE0;      //ID with 16 bits
//u32  ExdID=0xC6D99F88;  //ID with 32 bits
for(i==0;i<1;i++)
{
  if(i==0)
    //CAN with bandwidth of 500k/bps
    Init_CAN(can_500k);
  else
    //CAN with bandwidth of 250k/bps
    Init_CAN(can_250k);

  //acquire CAN frame with id=FD00
  CAN1_Config16BitFilter(0xfd00,0xfd00);

  //send Req frame
  iLen=Send_Frame_StCanEx
    (StdID,”\x02\x1\x0”,3,iAnscmd);
  if(iLen<=0){
    CAN1_Config16BitFilter(0xfd20,0xfd20);
    iLen=Send_Frame_StCanEx
      (StdID,”\x02\x1\x0”,3,iAnscmd);
    if(iLen<=0){  //if iLen<=0, try again
      SendFlag=1;
      CAN1_Config32BitFilter(ExdID,0XC6D78880);
      iLen=Send_Frame_StCanEx
        (ExdID,”\x02\x1\x0”,3,iAnscmd);
      if(iLen>0){
        //success
        SendFlag=1;
        break;
        }
      else SendFlag=0;
      }
    else{
      break; //success
      }
    }
  else{
    CAN1_Config16BitFilter(0xfd00,0xfd00);
    break;
  }
}
return iLen;
}
```

The main implementation codes about the LCD displayer are as follows:

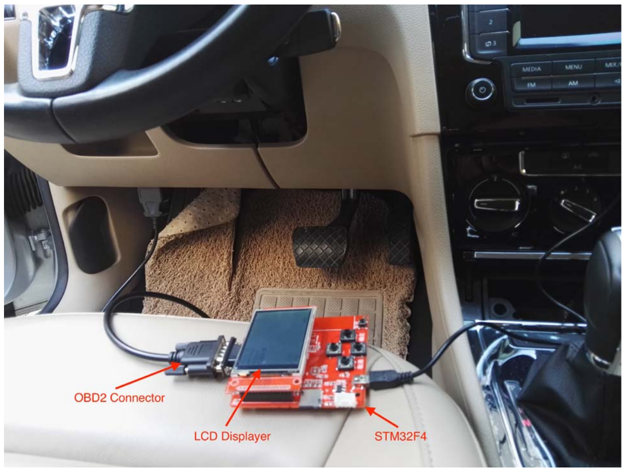


Fig. 4. The Installing of DAS in Real Vehicles

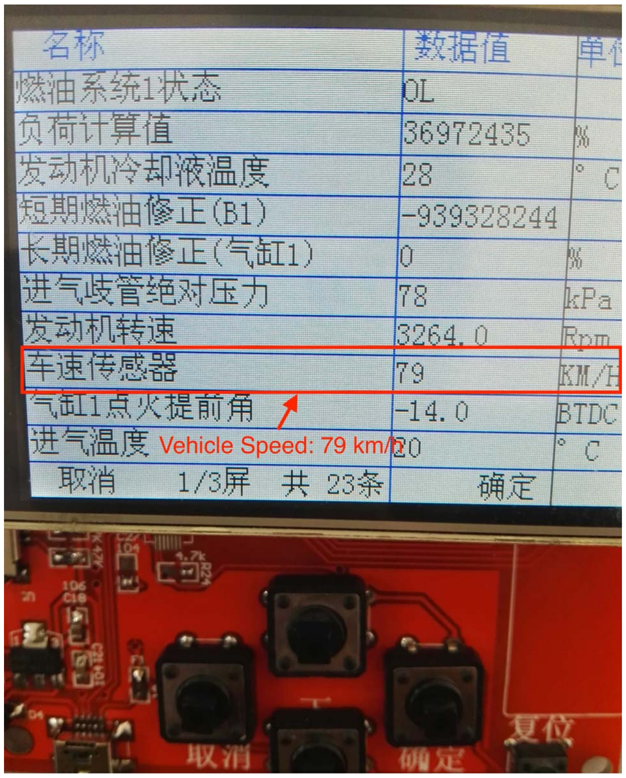
```
u8 InitMenu_Dtc(iMenu *p)
{
  unsigned char i=0;
  u8 key_val=0;
  int iMenuCount=0;
  KEY_Init();

  LcdShow_StrEx(15,220 ,”No”);
  LcdShow_StrEx(240,220 ,”Yes”);
  for(i=0;i<p->iCount;i++){
    POINT_COLOR=BLACK;
    LcdShow_StrEx(130,0 ,”<—”);
    POINT_COLOR=BLACK;
    LcdShow_StrEx(1,24*i,p->iShowMenu[i]);
    }

  while(1){
    key_val=KEY_Scan();
    if(key_val==Key_OK){
      return iMenuCount; }
    else if(key_val==Key_EXIT){
      return 0xff; }
    else if(key_val==Key_UP){ //up and down
      LcdShow_StrEx(130,24*iMenuCount ,”␣␣”);
      iMenuCount--;
      if(iMenuCount<0)iMenuCount=p->iCount-1;
      LcdShow_StrEx(130,24*iMenuCount ,”<—”);
      }
    else if(key_val==Key_DOWN){
      LcdShow_StrEx(130,24*iMenuCount ,”␣␣”);
      iMenuCount++;
      if(iMenuCount==p->iCount)iMenuCount=0;
      LcdShow_StrEx(130,24*iMenuCount ,”<—”);
    }
    delay_ms(100);
  }
}
```

IV. PROTOTYPE SYSTEM AND VERIFICATION

The implemented prototype of DAS is shown in Figure 4, and key components of DAS are labeled. We installed the DAS in a real vehicle, and acquired the vehicle data in real-time while the vehicle is driving as shown in Figure 5.



名称	数据值	单
燃油系统1状态	OL	
负荷计算值	36972435	%
发动机冷却液温度	28	° C
短期燃油修正(B1)	-939328244	
长期燃油修正(气缸1)	0	%
进气歧管绝对压力	78	kPa
发动机转速	3264.0	Rpm
车速传感器	79	KM/H
气缸1点火提前角	-14.0	BTDC
进气温度	20	° C

取消 1/3屏 共 23条 确定

Fig. 5. The Acquired Vehicle Data through DAS

V. Conclusion

Vehicle data acquisition is meaningful to the Internet-of-Vehicles, as different intelligent functions such as on-line diagnosis and security-attacking detection can be realized based on the acquired data, which can help to ensure the safe driving and improve the driving experience. As a result, we implemented a vehicle data acquisition system based on STM32, LED display and etc. By installing it on a real vehicle, we testified that it can get the vehicle data in real-time. In our future work, the acquired vehicle-data will be sent to the edge cloud server through the 4G/IPv6-based WIFI, and further analysis of it will be to realize the above mentioned intelligent functions.

Acknowledgment

The authors would like to express their gratitude to the anonymous reviewers for their constructive comments which have helped to improve the quality of the paper. This work was partially supported by Natural Science Foundation of China(61502405), Fujian Educational Bureau(JA15368), the CERNET Innovation Project(NGII20161003, NGII201610201,NGII20160708), the Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology(2016WLZC020), the Research Foundation of Education Bureau of Hunan Province(16B085), the Science and Technology Projects of Hunan Province(2016JC2074).

References

[1] R Coppola, M Morisio. Connected Car: Technologies, Issues, Future Trends. ACM Computing Surveys, 2016, 49(3): 46.
[2] F C Yang, S G Wang, J L Li, Z H Liu, Q B Sun. An overview of internet of vehicles. China Communications, 2014,11(11): 1-15.
[3] John Greenough. Connecting cars to the internet has created a massive new business opportunity. http://www.businessinsider.com/connected-car-market-forecast-report-2015-5. 2017.2.27.
[4] R N Charette. This Car Runs on Code. IEEE Spectrum, February 2009.
[5] S Tuohy, M Glavin, C Hughes, E Jones, M Trivedi, L Kilmartin. Intra-Vehicle networks: A review. IEEE Trans. on Intelligent Transportation Systems, 2015,16(2): 534-545.
[6] H F Zhang, W Kang. Design of the Data Acquisition System Based on STM32. Procedia Computer Science, 2013, 17: 222-228.
[7] M J Kang, J W Kang. A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security. In Proceedings of IEEE Vehicular Technology Conference, 2016, 1-5.
[8] J J Jain, C Busso. Analysis of driver behaviors during common tasks using frontal video camera and CAN-Bus information. In Proceedings of IEEE International Conference on Multimedia and Expo, 2011.
[9] X Hou, Y Li, M Chen, D Wu, D Jin, S Chen. Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures. IEEE Transaction on Vehicular Technology, 2016, 65(6): 3860-3873.
[10] STM32. https://en.wikipedia.org/wiki/STM32. 2017.2.24.
[11] On-Board Diagnosis. https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-34. 2017.2.24.
[12] ISO 15765. https://en.wikipedia.org/wiki/ISO_15765-2. 2017.2.27.