Development of a prototype dashboard for wheeled passenger transport

D Astapenko, D Butarovich and A Smirnov

Wheeled Vehicles Department of Bauman Moscow State Technical University, Moscow, 105005, Russian Federation

E-mail: astapenkoda@bmstu.ru

Abstract. The development of wheeled passenger transport is associated with the need to use many components from various manufacturers. All these components have parameters that should be displayed on the dashboard. In the modern world, the dashboard is the "face" of the vehicle. It should give the driver a complete picture of the vehicle condition. The information shown on the screen should be understandable and not misleading for the user, regardless of the level of his training. World trends show that the analog dashboard is being replaced by fully digital (graphical) ones. When the graphics is added, the software becomes considerably more complicated. In order to reduce the complexity of its development and to design it in parallel, a modular approach is used. For the dashboard, four software modules can be distinguished: drivers for working with the hardware, operating system, top layer software, and a graphical user interface. Modern dashboards use processors with operating systems based on Embedded Linux due to the fact that there are many ready-made drivers for the hardware for this operating system. To accelerate software development, European manufacturers use ready-made frameworks for working with graphics, for example, the cross-platform Qt framework. The article presents the results of the development of a prototype dashboard for wheeled passenger vehicles. The device components and necessary functionality of the product for mass production is determined.

1. Introduction

Currently, in Russia and in the world the development of the wheeled passenger electric transport: trolleybuses, buses, and electric buses, is under way [1,2,3,4,5]. A lot of attention in the development of the vehicles is given to the dashboard [6,7,8]. The dashboard is the part of the driver’s workplace which displays the basic parameters of the vehicle, as a result of which the driver constantly interacts with it. Thus, the dashboard is the “face” of the entire vehicle. At the prototype stage, when the vehicle’s functionality is not fully defined, the display of parameters for their control plays a large role. To do this, one must constantly make changes to the dashboard. The article presents the result of using a dashboard development method that solves this problem.

2. Type and composition of the dashboard

The dashboard can be of 3 types: analog (figure 1a), digital (graphic) (figure 1b) and combined (figure 1c).

The analog dashboard is a set of analog indicators and indicator lights. The digital dashboard is a fully graphical display. The combined dashboard, like the analog one, consists of arrow indicators and
indicator lights, but it also uses a small digital display. Comparison of dashboard types is given in table 1.

| Type   | Advantages                           | Disadvantages                           |
|--------|--------------------------------------|-----------------------------------------|
| Analog | low cost, low complexity software, low complexity hardware | low versatility, impossible to change |
| Digital| high versatility, easy to change     | high cost, high complexity software, high complexity hardware |
| Combined| middle cost, low complexity hardware | middle versatility, middle software, only the digital part can be changed |

At the moment, leading manufacturers of vehicles use fully digital displays. This is due to the fact that:

1) the power of computing devices (microprocessors) has increased significantly, and their cost has decreased,
2) a large number of software tools appeared - frameworks that reduce the complexity of software development,
3) consumer desire to own a car with a graphic display. Due to the mass introduction of smartphones with graphic displays and customization options for users, users are accustomed to extensive personalization options,
4) the graphical interface is a more flexible solution - by changing the software, you can quickly adapt the dashboard to a new vehicle.

![Figure 1. Dashboards: a – analog, b – digital, c – combined.](image)

Next, the results of the dashboard development for wheeled passenger transport will be presented. The development of a dashboard with a graphical interface includes the following subtasks:

1) collecting data from different communication interfaces,
2) providing different displays for different vehicles, 
3) development of different graphical interfaces for different vehicle models, 
4) development of universal top layer software that does not depend on the hardware of the device, 
5) quick adaptation of the interface to new vehicle components, 
6) hardware development.

Software development for a dashboard is a complex task. To reduce the complexity of the development and to provide development in parallel, a modular approach is used. The software structure used in developing the prototype dashboard for a wheeled passenger vehicle is shown in figure 2.

![Figure 2. Dashboard software structure.](image)

For the dashboard, four software modules can be distinguished: drivers for working with the hardware, operating system, top layer software, and a graphical interface. Modern dashboards use processors with Linux-based operating systems due to the fact that there are many ready-made drivers for the hardware for this operating system and it does not require large computational resources. To speed up software development, many manufacturers use ready-made frameworks for working with graphics, one of them is the cross-platform Qt framework. It provides ease of interconnection between the operating system, top layer software and a graphical interface.

### 3. Hardware

Developing hardware from scratch for a prototype product is economically unjustified. It is more cost-efficient to use ready-made modules and link them together, and when the product is ready for mass production, make new hardware based on the modules used. The hardware of the dashboard is an assembly of three modules: single-board computer Raspberry Pi 3 B+, a display controller board, and a proprietary board (figure 3). The Raspberry Pi single-board computer is used as the computing center of the device, and the software being developed is running on it. The display controller board is used to convert the HDMI interface to LVDS, and to adjust the display settings (contrast, screen calibration, etc.).

The proprietary development board expands the functionality of the Raspberry Pi 3 B+, as it does not have enough peripherals for the developed dashboard. Proprietary elements are located on the board: a voltage converter, an interface for connecting to the CAN bus, a real-time clock, elements for connecting control buttons, a display backlight control module. The 3D model of the board is shown in figure 4.
4. Operation system and hardware drivers

A special Linux build using the Yocto project is used as the operating system. The Yocto Project is an open source collaboration project that helps engineers to create custom Linux-based systems for embedded products, regardless of the hardware architecture. The project provides a standard delivering hardware support and software stacks, allowing the interchange of software configurations and builds. The tools allow users to build and support customizations for multiple hardware platforms and software stacks in a maintainable and scalable way. The Yocto Project combines, maintains and validates three key development elements (figure 5) [9]:

1) the set of integrated tools to make working with embedded Linux successful, including tools for automated building and testing, processes for board support and license compliance, and component information for custom Linux-based embedded operating systems,
2) the reference embedded distribution (called Poky),
3) the OpenEmbedded build system, co-maintained with the OpenEmbedded Project.

Hardware drivers contain in the Board Support Package. A Board Support Package (BSP) is the collection of information that defines how to support a particular hardware device, set of devices, or hardware platform. The BSP includes information about the hardware features present on the device and kernel configuration information along with any additional hardware drivers required. The BSP also lists any additional software components required in addition to a generic Linux software stack for both essential and optional platform features [10].
5. Top layer software
Dashboard software was developed in several programming languages. The control logic uses the C++ language with the Qt framework. Algorithms that are responsible for receiving and sending CAN messages, calculating mileage, estimated range, etc. are written in C++. To work with graphics, the qml language using the QtQuick module is used. Animation of arrow movements, font size changes, display color changes is written in qml. Python is used to automatically generate code from a .dbc file for working with CAN messages (figure 6).

![Diagram](image6.png)

**Figure 6.** Programming languages used in the project.

Most often, the CAN bus is used to communicate units in a vehicle. A file with the .dbc extension is used to describe messages transmitted over the bus. It contains a textual description of CAN signals and frames. To reduce the adaptation time of the dashboard to new vehicles, and minimize errors when writing code, a script was developed that allows you to create .cpp and .h files from a .dbc file for use in a C++ project (figure 7).

![Diagram](image7.png)

**Figure 7.** Generating C++ code from .dbc file.

The basic logic of the dashboard is developed in C++ using the objects of the Qt framework classes, the use of these objects makes the software more universal.

To work with the graphical interface, the Qml language and the QtQuick module are used. QtQuick can work with most modern graphic formats. Source images can be created in any specialized application. The image is exported to QtQuick as special objects. Work with graphics changes the properties of these objects.

To configure the transfer of objects from QML to C++, and vice versa, the Signals and slots mechanism is used (figure 8). Signals and slots are used for communication between objects.

![Diagram](image8.png)

**Figure 8.** Mechanism Signals and slots.
The signals and slots mechanism is a central feature of Qt. A signal is emitted when a particular event occurs. Qt's widgets have many predefined signals, custom signals can also be added. A slot is a function that is called in response to a particular signal. Qt's widgets have many pre-defined slots, custom slots can also be added [11].

6. Graphical user interface

The dashboard screen can be divided into several functional parts. The dials are shown on the left and right sides. On the left side the following elements are shown: 3 dial indicators (ammeter, pressure in the first brake circuit, pressure in the second brake circuit) and temperature scale, temperature outside and inside the passenger compartment. On the right side there is one dial indicator (speed), fuel gauge, mileage and current date. On the central part the following elements are located: door indication, critical errors, a field for displaying text information and the current time.

![Figure 9. Dashboard interface.](image)

If there are no errors in the system while the vehicle is moving, the logo is displayed on the central part (figure 10). The upper part of the display is reserved for status lights.

![Figure 10. Dashboard interface, without error.](image)

In the future, it is planned to add support for several interface options so that the user can choose it to his liking.

7. Additional configuration for run Dashboard application

To use the device as a dashboard, some additional settings need to be made.

Before starting the main application, you need to start the CAN interface. There are two ways to do this. The first way is to use the QProcess class. To execute a command, it should be passed to the start method and wait for completion:

```cpp
QProcess process;
process.start("Custom command");
process.waitForFinished();
```
The second way is to run a custom script before running the application. To do that edit the:

```
/lib/systemd/system/b2qt.service
```

Under [Service] section you can add:

```
ExecStartPre=<your_custom_command>.
```

### 8. Dashboard testing

A special test bench is used to test the dashboard. The bench is assembled on the debug board from the real peripheral modules (figure 11). By testing one can check the following: the correctness of the device interaction with the CAN bus, conformance of the displayed data with the DBC file, reading and writing date/time from the RTC module, backlight control.

![Figure 11. Dashboard test bench.](image)

Using cross-compilation, the project can be assembled on the host computer and only the executable file can be send to the device. Due to this development approach, you can quickly make changes to the algorithms of the device.

### 9. Conclusion

The results of bench tests of the dashboard have confirmed the workability of using a prototype device on a vehicle. The modular approach to software development has allowed:

1) reduction of the development time, development from scratch to a working prototype took 3 months,
2) achieving software independence from the hardware,
3) using various graphical interfaces.
Further, it is planned to increase the application loading speed from a completely switched off state, now the launch time is from 5 to 7 seconds. In normal mode, it is planned to display more useful information on the central part of the screen: a map with GPS navigation when Drive is in the motion mode, or a rear view camera when reversing.

References
[1] Astapenko D A, Butarovich D O and Smirnov A A 2019 Development of the upper-layer electronic control unit based on a production-type unit for a commercial vehicle *IOP Conf. Series: Materials Science and Engineering* **534**

[2] Buzunov N V, Kotiev G O and Gorelov V A 2019 Implementation of the interaction of the steering wheel loader control system of the remote-controlled wheeled vehicle operator interface with a real-time simulation model *IOP Conf. Series: Materials Science and Engineering* **534**

[3] Buzunov N V, Kotiev G O and Padalkin B V 2019 Wheel vehicle dynamics real-time simulation for on-board stand-alone moving control system realization *IOP Conf. Series: Materials Science and Engineering* **534**

[4] Gorelov V A, Butarovich D O, Staduhin A A and Skotnikov G I 2019 Simulation-based identification of the parameters of a minibus hybrid powertrain *IOP Conf. Series: Materials Science and Engineering* **534**

[5] Kotiev G O, Miroshnichenko A V, Stadukhin A A and Kositsyn B B 2019 Estimating operation modes for the individual wheel electric drive of the all-wheel drive vehicle with the use of the driving simulator *IOP Conf. Series: Materials Science and Engineering* **534**

[6] Muslim E, Moch B N, Lestari R A, Shabrina G and Ramardhiani R 2019 Ergonomic design of electric vehicle instrument panel: a study case on Universitas Indonesia’s national electric car *IOP Conf. Series: Materials Science and Engineering* **508**

[7] Pawlenka T, Kulhanek J, Tomcik P and Zapletal R 2019 Design of Digital CAN Based Car Dashboard Unit *Proceedings of the 2019 20th International Carpathian Control Conference*

[8] Ren H, Tan Y and Zhang N 2019 Research on Form Design of Automotive Dashboard Based on Kansei Engineering *IOP Conf. Series: Materials Science and Engineering* **573**

[9] Software – Yocto Project. Yocto Project. Retrieved September 14, 2019, from https://www.yoctoproject.org/software-overview

[10] Yocto Project Board Support Package Developer’s Guide. Yocto Project. Retrieved September 14, 2019, from https://www.yoctoproject.org/docs/2.7.1/bsp-guide/bsp-guide.html#bsp

[11] Signals & Slots | Qt Core 5.13.1. Qt. Retrieved September 14, 2019, from https://doc.qt.io/qt-5/signalsandslots.html

Acknowledgments
The research is done in BMSTU with the financial support of the Ministry of Education and Science of Russian Federation under the Agreement № 14.574.21.0178 (Unique identifier: RFMEFI57417X0178).