Votary Ultimate Cluster 2K

Next-Generation Smart Instrument Cluster for Embedded Android

Target Platform: Forlinx OKT507-C (Android 10)

Framework: Qt 5.15.2 (C++ & QML)

1. Introduction

The **Votary Ultimate Cluster 2K** is a cutting-edge digital instrument cluster designed for modern electric 2-wheelers. Unlike traditional analog dashboards, this system provides a centralized interface for vehicle telemetry, navigation, and smart connectivity.

The system runs on the **OKT507-C Single Board Computer** (Allwinner H616) with **Android 10**. It acts as the primary **Android Launcher**, meaning it boots directly into the dashboard interface, providing an instant-on, automotive-grade user experience.

1.1 Project Objectives

- **Digital Telemetry:** Real-time display of Speed, Temperature, Humidity, and Vehicle Tilt.
- **Smart Rider Connectivity: Bluetooth Call & SMS Notifications** displayed directly on the cluster for safety.
- **Navigation:** Integrated Google Maps via Android WebView.
- **Cloud IoT:** Remote data synchronization using Google Firebase.
- **Security:** PIN-based Lock Screen authentication.

2. Hardware Architecture

The system utilizes a distributed architecture consisting of a **Sensor Node** and a **Display Unit**.

2.1 The Sensor Node (ESP32)

The ESP32 microcontroller acts as the Vehicle Control Unit (VCU) simulator:

- **Throttle Simulation:** A potentiometer maps voltage changes to speed (0–120 km/h).
- **Temperature Monitoring:** The internal temperature sensor of the MPU6050 is used to obtain temperature readings. This sensor provides sufficient accuracy for system-level monitoring in a prototype environment.
- **Vehicle Dynamics:** An MPU6050 Accelerometer/Gyroscope captures vehicle pitch and roll (Tilt).

2.2 The Display Unit (OKT507-C)

- **Processor:** Allwinner H616 Quad-core Cortex-A53.
- **OS:** Android 10 (Custom ROM).
- **Role:** Runs the Qt Application, manages Bluetooth/Wi-Fi stacks, and renders the HMI (Human-Machine Interface).

3. Communication Protocols

We engineered a multi-channel communication system to ensure robust data delivery.

3.1 Cloud Sync (Firebase Realtime Database)

The ESP32 pushes sensor data to Google Firebase via Wi-Fi. The Dashboard fetches this data via HTTP GET requests. This allows the vehicle status to be monitored remotely from anywhere in the world.

3.2 Bluetooth Smart Notifications

The OKT507-C pairs with the rider's smartphone. Using the Android Bluetooth API, the cluster intercepts incoming **Call States** (Ringing/Off-hook) and **SMS Intents**.

- **Feature:** When a call comes in, the dashboard displays the caller's name/number.
- **Benefit:** Increases rider safety by reducing the need to check the phone while riding.

3.3 Direct USB Serial & Local Wi-Fi

- **USB:** Physical connection via USB-UART (`/dev/ttyUSB0`) for zero-latency data transfer.
- **Local Wi-Fi:** Socket-based TCP/UDP communication for high-speed local telemetry when offline.

4. Software Implementation

The application follows the **Model-View-Controller (MVC)** design pattern using Qt 5.15.

4.1 Technologies Used

- **Qt/QML:** For high-performance, fluid UI animations (Speedometer, Transitions).
- **C++:** For backend logic, Serial Port management, and hardware interfacing.
- **Android Java/JNI:** For accessing native Android features (Bluetooth adapter, WifiManager, Toast notifications).
- **Google Maps API:** Embedded via `QtWebView`.

4.2 Key Software Components

1. **`main.qml`:** The UI Entry point. Manages the "StackView" (Dashboard, Maps, Settings).
2. **`WifiManager.cpp`:** Scans for available networks and reports signal strength.
3. **`AndroidManifest.xml`:** Configures permissions (`BLUETOOTH`, `INTERNET`) and sets the app as the **System Launcher** (`android.intent.category.HOME`).
4. **`SimpleDemo.pro`:** The build configuration file linking `androidextras`, `network`, and `bluetooth` modules.
5. **`main.cpp`:** Initializes the Qt application, loads the QML UI, and connects C++ backend logic (sensor, Wi-Fi, serial data) to the frontend using signals and slots.

5. Challenges & Lessons Learned

5.1 The SSL Certificate Issue

- **Challenge:** The OKT507 board often reset its system time to 1970 upon reboot. This caused secure HTTPS requests to Firebase to fail due to certificate invalidity ("SSL Handshake Error").
- **Solution:** We implemented a fallback mechanism to use **HTTP (Non-Secure)** requests when SSL fails, ensuring data continuity regardless of the system date.

5.2 Android Network Caching

- **Challenge:** The Android network stack aggressively cached the JSON data from Firebase. The dashboard would show the same speed value indefinitely.
- **Solution:** We implemented "Cache Busting" by appending a unique timestamp to every URL request (`?t=timestamp`), forcing the network to fetch fresh data every second.

5.3 Build & Deployment

- **Challenge:** Complexity in cross-compiling Qt for the ARMv7 architecture led to "Binary Not Found" errors during deployment.
- **Solution:** We created a standardized "Clean Build Script" that resets the build environment and correctly paths the Android SDK/NDK before every flash.

---

6. Conclusion

The **Votary Ultimate Cluster 2K** successfully demonstrates the integration of embedded hardware, mobile operating systems, and cloud connectivity. By combining the **ESP32's** sensor capabilities with the **OKT507-C's** powerful Android interface, we created a dashboard that is not only a display but a central intelligence hub for the rider.

The addition of **Bluetooth Notifications** and **Cloud Telemetry** brings this prototype to a commercial-ready standard, offering features comparable to high-end EVs currently in the market.