Automotive Dashboard AGI System

An advanced AI-powered vehicle dashboard assistant that provides real-time monitoring, analysis, and interaction through OBD-II diagnostics, computer vision, GPS navigation, and natural language conversation.

🚗 Features

Core Capabilities

- 🔧 **OBD-II Diagnostics**: Real-time vehicle health monitoring
  - Engine RPM, speed, temperature
  - Fuel level, battery voltage
  - Diagnostic trouble codes (DTCs)
  - Proactive alerts for critical conditions
- 👁️ **Computer Vision**: Continuous road monitoring via dashcam
  - Object detection (vehicles, pedestrians, cyclists, animals)
  - Hazard identification
  - Traffic density assessment
  - Weather condition analysis
  - Automated frame capture every 3 minutes
- 🗺️ **GPS Navigation**: Intelligent routing and location services
  - Real-time location tracking
  - Turn-by-turn directions
  - Route optimization
  - Address geocoding
- 🤖 **Conversational AGI**: Natural language interaction
  - Context-aware responses
  - Proactive safety alerts
  - Vehicle diagnostics interpretation
  - Navigation assistance
  - General queries and conversation

Safety Features

- ⚠️ Real-time hazard detection
- 🌡️ Engine temperature monitoring
- 🔋 Battery health alerts
- ⛽ Fuel level warnings

- 🚦 Traffic condition awareness
- 👥 Pedestrian and cyclist detection

🏗️ Architecture

```
DashboardAGI
├── OBDInterface        # Vehicle diagnostics via OBD-II port
├── DashcamVision       # YOLO-based object detection & scene analysis
├── NavigationService   # GPS location & OSRM routing
├── AudioHandler        # Speech-to-text & text-to-speech
├── AGIBrain            # OpenAI GPT integration with full context
└── Background Threads  # Continuous monitoring (OBD + Vision)
```

📋 Prerequisites

Hardware Requirements

- **Computer/Raspberry Pi** with:
  - USB camera or dashcam (720p+ recommended)
  - Microphone
  - Audio output (speaker)
  - USB OBD-II adapter (ELM327 or similar)
- **Vehicle**:
  - OBD-II port (standard in cars post-1996)

Software Requirements

- Python 3.8+
- Linux/Unix (recommended) or Windows
- Audio playback utility (`mpg123` for Linux)

🚀 Installation

1. Clone Repository

```bash
git clone <repository-url>
cd dashboard-agi
```

2. Create Virtual Environment

```bash
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Dependencies

```bash
pip install -r requirements_agi.txt
```

4. Install System Audio Player

Linux:

```bash
sudo apt update
sudo apt install mpg123
```

macOS:

```bash
brew install mpg123
```

Windows: Audio will play automatically via system default player.

5. Configure Environment

```bash
cp .env.example .env
nano .env  # Add your OpenAI API key
```

Required environment variable:

```bash
OPENAI_API_KEY=sk-proj-xxxxxxxxxxxxx
```

6. Download YOLO Model

The YOLOv8 nano model will download automatically on first run, or manually:

```bash
```

```
wget https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt
```

⚙️ Configuration

Environment Variables (.env)

```bash
# Required
OPENAI_API_KEY=your_openai_api_key_here

# Optional Overrides
CAMERA_INDEX=0
OBD_PORT=/dev/ttyUSB0
VISION_ANALYSIS_INTERVAL=180
OBD_POLL_INTERVAL=2
```

JSON Configuration (config.json)

```json
{
  "CAMERA_INDEX": 0,
  "VISION_ANALYSIS_INTERVAL": 180,
  "OBD_ENABLED": true,
  "OBD_PORT": "/dev/ttyUSB0",
  "SAVE_ANALYZED_FRAMES": true,
  "ENABLE_VOICE_ALERTS": true,
  "LLM_MODEL": "gpt-4-turbo-preview"
}
```

🎯 Usage

Basic Operation

```bash
python dashboard_agi.py
```

The system will:

1. Initialize all subsystems (OBD, camera, audio, AI)
2. Start background monitoring threads
3. Greet you with "Dashboard AGI online..."
4. Listen for voice commands continuously

Voice Commands Examples

Navigation:

- "Take me to Pietermaritzburg City Hall"
- "What's the fastest route home?"
- "How far to my destination?"

Vehicle Status:

- "What's my fuel level?"
- "Check engine temperature"
- "Are there any warning codes?"
- "Give me a full vehicle status"

Road Conditions:

- "What do you see ahead?"
- "Any hazards nearby?"
- "How's traffic?"

General:

- "Tell me a joke"
- "What's the weather like?"
- "Remind me to check tire pressure"

Exit:

- "Shutdown" / "Exit" / "Quit"

📊 Data Output

File Structure

```
dashboard_data/
├── dashboard_agi.log        # Main system log
├── obd_data.jsonl           # OBD-II data history (JSONL format)
├── camera_frames/           # Captured dashcam frames
│   ├── frame_20240101_120000.jpg
│   └── ...
├── driver_voice.wav         # Last recorded voice input
└── assistant_response.mp3   # Last TTS output
```

Log Levels

- **DEBUG**: Detailed system operations
- **INFO**: Key events and actions
- **WARNING**: Non-critical issues
- **ERROR**: System errors
- **CRITICAL**: Safety-critical events

🔒 Security Best Practices

API Keys

- ✅ Store in `.env` file (gitignored)
- ✅ Never commit to version control
- ✅ Use environment variables in production
- ❌ Don't hardcode in source files

OBD-II Access

- The OBD port provides direct access to vehicle systems
- Only use trusted OBD-II adapters
- Disconnect when not in use

Privacy

- Camera frames are saved locally
- No data is transmitted except to configured APIs
- Review and delete old frames periodically

🐛 Troubleshooting

Camera Issues

```python
# Test camera
python -c "import cv2; cap = cv2.VideoCapture(0); print(cap.isOpened())"
```

Try different camera indices (0, 1, 2...) in config.

OBD-II Connection

```bash
```

```
# Check USB device
ls -l /dev/ttyUSB*

# Test with screen
screen /dev/ttyUSB0 38400
```

If no OBD adapter, system will use simulated data for testing.

Audio Issues

```bash
# Test microphone
python -c "import sounddevice as sd; print(sd.query_devices())"

# Test speaker
mpg123 test.mp3
```

OpenAI API Errors

- Verify API key is valid
- Check account has credits
- Ensure internet connectivity
- Review rate limits

🔧 Development

Running in Debug Mode

```python
# Modify config for verbose logging
config = DashboardConfig()
logger.setLevel(logging.DEBUG)
```

Testing Without Hardware

The system includes simulated data for:

- OBD-II (when no adapter connected)
- Camera (when no device available)

Custom Extensions

Add custom skills by extending the `AGIBrain` class:

```python
def process_custom_command(self, command: str) -> str:
    if "maintenance" in command:
        return self.generate_maintenance_schedule()
    return self.process_input(command)
```

📈 Performance Optimization

Resource Usage

- **CPU**: ~15-30% (single core) during normal operation
- **RAM**: ~500MB-1GB
- **Storage**: ~1GB per week (with frame saving)

Optimization Tips

1. Reduce vision analysis frequency (increase `VISION_ANALYSIS_INTERVAL`)
2. Lower camera resolution in config
3. Use YOLOv8n (nano) model for speed
4. Disable frame saving if storage limited
5. Reduce OBD polling frequency for older systems

🛠️ Hardware Setup Guide

OBD-II Adapter Connection

1. **Locate OBD-II port** (usually under steering wheel)
2. **Plug in ELM327 adapter** (Bluetooth or USB)
3. **For USB**: Will appear as `/dev/ttyUSB0` (Linux) or `COM3` (Windows)
4. **For Bluetooth**: Pair device first, then connect

Camera Mounting

- Mount dashcam with clear forward view
- Avoid windshield obstructions
- Ensure stable mounting (vibration resistant)
- Connect via USB to computer

Audio Setup

- Use high-quality microphone for voice recognition

- Position speaker for clear audio in vehicle
- Test in actual driving conditions

📚 API Documentation

OpenAI Integration

Model: GPT-4 Turbo (configurable)

- Context window: 128K tokens
- Temperature: 0.7 (balanced creativity/accuracy)
- Max tokens per response: 800

OSRM Routing

- Public instance: `http://router.project-osrm.org`
- For production: Consider self-hosted instance
- Rate limits apply to public instance

🤝 Contributing

Contributions welcome! Areas for improvement:

- [ ] Enhanced OBD-II protocol support
- [ ] Advanced vision algorithms (lane detection, traffic lights)
- [ ] Multi-language support
- [ ] Mobile app integration
- [ ] Cloud sync capabilities
- [ ] Voice wake word detection
- [ ] Offline AI models

📄 License

MIT License - see LICENSE file

⚠️ Disclaimer

This system is designed to assist drivers, not replace their judgment and attention. Always:

- Keep eyes on the road
- Obey all traffic laws
- Use system only when safe
- Regularly maintain your vehicle
- The system is for informational purposes only

🙏 Acknowledgments

- OpenAI for GPT models
- Ultralytics for YOLO
- OpenStreetMap & OSRM for routing
- Python OBD library maintainers

📞 Support

For issues, questions, or feature requests:

- Open an issue on GitHub
- Check troubleshooting section
- Review logs in `dashboard_data/`

Drive Safe! 🚙 💨