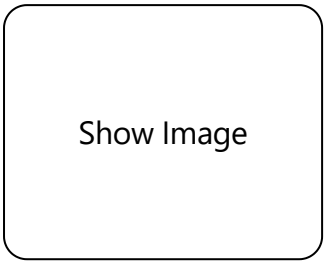
ANPR System for Orange Pi 5 Ultra

A modular, real-time **Automatic Number Plate Recognition (ANPR) system** designed specifically for the **Orange Pi 5 Ultra** hardware.



🚀 Features

- **Hardware-Optimized Detection**:
  - Utilizes RK3588 NPU (6 TOPS) via RKNN for maximum performance
  - Falls back to Mali-G610 MP4 GPU via OpenCL/OpenGL when NPU not available
  - Efficient CPU fallback for compatibility
- **License Plate Detection**:
  - YOLO11s model for accurate detection
  - Auto-selection of best available hardware
- **Enhanced OCR**:
  - Multiple preprocessing methods for optimal results
  - Skew correction via projection profiles for angled plates
  - Text normalization (O→0, I→1, etc.)
- **Vehicle Attribute Detection**:
  - Optional vehicle make recognition using ResNet18
  - Optional vehicle color detection using K-means clustering
- **Access Control**:
  - Validation against allowlist
  - Optional make/color verification
  - GPIO trigger (pin 17) for gate control when matched
- **Reliability**:
  - Camera reconnection handling
  - Robust logging system
  - Hardware graceful fallback

🖥️ Target Hardware

- **Device**: Orange Pi 5 Ultra
- **CPU**: Cortex-A76 + A55 (8-core big.LITTLE)
- **GPU**: Mali-G610 MP4
- **NPU**: RK3588 6 TOPS (supports RKNN model format)
- **Operating System**: `Orangepi5ultra_1.0.0_ubuntu_jammy_desktop_gnome_linux5.10.160`

📋 Requirements

Hardware

- Orange Pi 5 Ultra
- CSI/USB Camera or RTSP IP Camera
- Optional: GPIO-controlled gate/barrier

Software

- Python 3.8+
- OpenCV 4.5+
- PyTorch (for GPU/CPU support)
- RKNN Toolkit Lite (for NPU support)
- EasyOCR
- scikit-learn (for color detection)

🔧 Installation

1. **Clone the repository**:

```bash
git clone https://github.com/yourusername/anpr-orangepi5ultra.git
cd anpr-orangepi5ultra
```

2. **Run the setup script**:

```bash
chmod +x setup.sh
sudo ./setup.sh
```

3. **Download model files**:
   - Place YOLO11s.pt in the `anpr_system/models/` directory for GPU/CPU detection
   - Place YOLO11s.rknn in the `anpr_system/models/` directory for NPU acceleration

- Place resnet18_vehicle_make.pth in the `anpr_system/models/` directory for vehicle make detection

4. **Configure your allowlist**:
   - Edit `anpr_system/allowlist.txt` to add authorized license plates
   - Optional: Edit `anpr_system/allowlist.json` for detailed vehicle information

🏃 Usage

Basic Usage

Run the ANPR system with:

```bash
cd anpr_system
python3 main.py --rtsp-url=rtsp://username:password@camera-ip:554/stream --show-video
```

Command Line Options

```
usage: main.py [-h] [--rtsp-url RTSP_URL] [--resolution RESOLUTION] [--enable-color]
               [--enable-make] [--allowlist ALLOWLIST] [--save-dir SAVE_DIR]
               [--log-dir LOG_DIR] [--show-video] [--show-debug]
               [--simulate SIMULATE] [--mock-gpio]

ANPR System for Orange Pi 5 Ultra

optional arguments:
  -h, --help            show this help message and exit
  --rtsp-url RTSP_URL   RTSP stream URL
  --resolution RESOLUTION
                        Stream resolution (480p, 720p, 1080p, or WIDTHxHEIGHT)
  --enable-color        Enable vehicle color detection
  --enable-make         Enable vehicle make detection
  --allowlist ALLOWLIST
                        Path to allowlist file
  --save-dir SAVE_DIR   Directory to save captured plates
  --log-dir LOG_DIR     Directory for log files
  --show-video          Show video window
  --show-debug          Show debug information
  --simulate SIMULATE   Use simulation images from directory instead of camera
  --mock-gpio           Use mock GPIO (for testing)
```

Testing

To test if all components are working correctly:

```bash
python3 test_system.py --enable-color --enable-make
```

For development or testing without a real camera:

```bash
python3 main.py --simulate=simulation_images --show-video --show-debug
```

Running as a Service

To install as a system service:

```bash
sudo cp anpr-system.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable anpr-system.service
sudo systemctl start anpr-system.service
```

🗂️ Project Structure

```
anpr_system/
├── main.py                # Entry point script
├── detectors/
│   ├── __init__.py
│   ├── yolo11_gpu.py      # YOLO11s using GPU (OpenCL-accelerated PyTorch)
│   └── yolo11_rknn.py     # YOLO11s via RKNN for NPU acceleration
├── vision/
│   ├── __init__.py
│   ├── ocr.py             # Enhanced OCR with preprocessing
│   └── skew.py            # Skew correction utilities
├── input/
│   ├── __init__.py
│   └── camera.py          # RTSP/OpenCV camera handler with reconnection
├── utils/
│   ├── __init__.py
│   ├── hardware.py        # Hardware detection: RKNN, GPU, CPU fallback
│   ├── logger.py          # Rotating log manager
│   ├── plate_checker.py   # Allowlist validation + GPIO control
│   ├── vehicle_make.py    # ResNet18 classifier for car make
│   ├── vehicle_color.py   # KMeans-based color detector
│   └── gpio.py            # Orange Pi-compatible GPIO control
├── models/                # YOLO11s.pt, YOLO11s.rknn, resnet18.pth, etc.
├── allowlist.txt          # Plate list (plain text)
├── allowlist.json         # Optional JSON with make/color verification
├── captures/              # Folder for cropped plate images
└── logs/                  # Log files
```

🔍 How It Works

System Flow

1. **Camera Input**: Captures frames from RTSP stream
2. **Object Detection**: Detects license plates using YOLO11s
3. **Plate Processing**:
   - Crops license plate region
   - Applies skew correction
   - Performs OCR with preprocessing
4. **Optional Analysis**:
   - Detects vehicle make
   - Detects vehicle color
5. **Validation**:

- Checks plate against allowlist
- Validates make/color if enabled

6. **Actions**:
   - Triggers GPIO for gate control if matched
   - Logs detection details
   - Saves cropped plate images

Hardware Utilization

The system automatically selects the best available hardware:

1. **NPU (First Choice)**:
   - Uses RKNN for YOLO model execution
   - Requires RKNN model format
2. **GPU (Second Choice)**:
   - Uses OpenCL-accelerated PyTorch
   - Accelerates OpenCV operations when possible
3. **CPU (Fallback)**:
   - Compatible with all components
   - Used when no acceleration is available

📊 Performance

Performance metrics on Orange Pi 5 Ultra:

Hardware	Resolution	FPS (Detection Only)	FPS (Full Pipeline)
NPU	720p	~25	~20
GPU	720p	~15	~12
CPU	720p	~5	~3

📝 License

👥 Contribution

Contributions are welcome! Please feel free to submit a Pull Request.

📞 Support

If you encounter any problems or have questions, please open an issue on the GitHub repository.