System User Manual – Wrong Way Driving Monitoring System (WWDMS)

System Name: Wrong Way Driving Monitoring System
Target Users: Traffic Officers, System Admins, City Transport Authorities
Institution: Mbeya University of Science and Technology
Developer: Elias Mang'era Mwita
Platform: Embedded (Arduino) + Desktop (Python-based GUI)
Date: July 2025

1. Overview

The **Wrong Way Driving Monitoring System (WWDMS)** is a hybrid system combining embedded hardware and software to automatically detect vehicles driving in the wrong direction. The system captures evidence, triggers alerts, and stores incident data for future reference. This helps enforce road rules and reduce accidents due to wrong-way driving.

System Components and Their Functions:

- **Arduino Prototype:** Includes sensors to detect vehicle movement, a camera module for image capture, a buzzer for audio alerts, and LEDs for visual warnings.
- **Python Simulation Module:** Uses OpenCV and machine learning to process live or recorded video streams and identify violations automatically.
- **Database & Logging:** Stores violation events, timestamps, images, and other metadata for administrative review and analysis.
- **Simulation Platform (Proteus 8.17):** Used during development to test electronic connections and circuit behavior virtually before hardware implementation.

2. Getting Started

2.1 Hardware Setup

Purpose: To prepare and activate the physical components for real-world detection.

Steps:

1. **Connect Components:** Link the camera, buzzer, LEDs, and sensors to the Arduino board based on the circuit diagram (see *Figure 3: Prototype Using Proteus*). Each component plays a critical role in detection and response.
2. **Upload Arduino Sketch:** Use the Arduino IDE to upload the embedded C/C++ code (called a sketch) to the board. This code controls how the hardware reacts to wrong-way movements.
3. **Power the System:** Use a USB cable connected to a PC or an external battery pack to supply power.

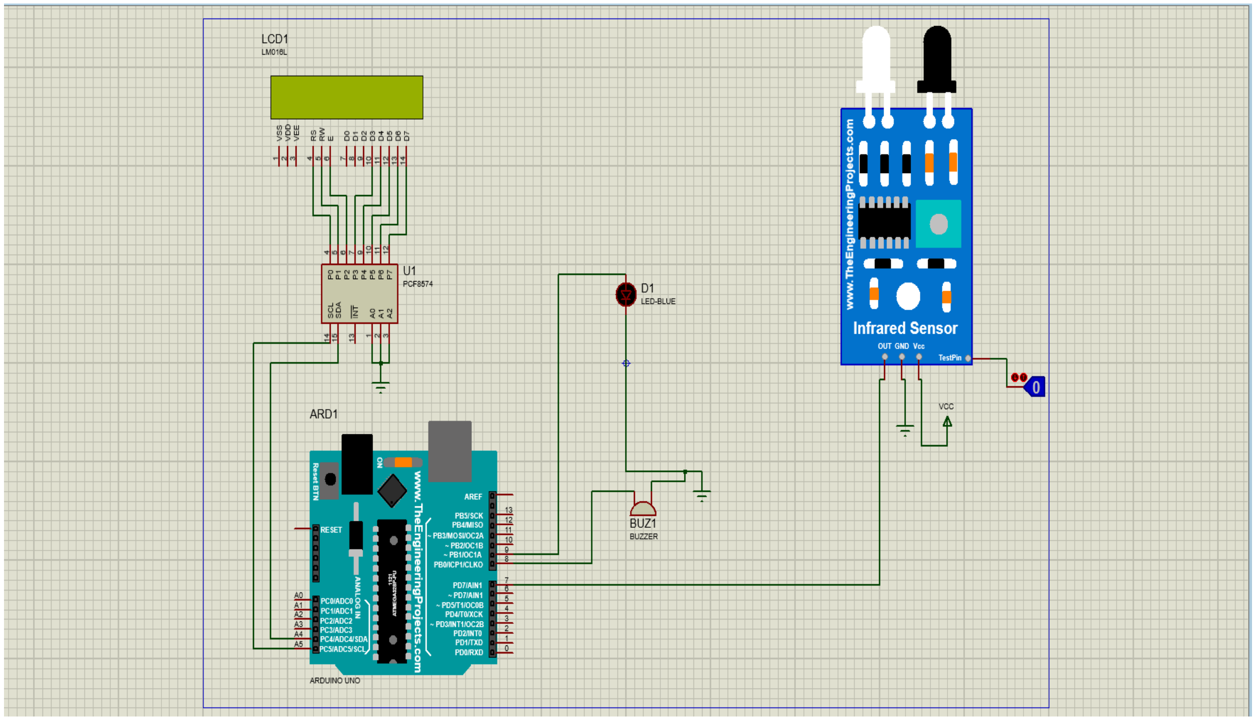

Figure 3: Prototype Using Proteus

2.2 Python Simulation (PC Application)

Purpose: To simulate the software part for detection using machine learning on test or live video.

Steps:

1. **Open Terminal/Command Prompt:** Launch your terminal to execute the script.
2. **Navigate to Project Directory:** Use the cd command to move into the folder containing your main Python script.
3. **Run the Python Script:** Execute the command:
4. python main.py

This script loads the trained ML model and starts analyzing the video feed.

5. **Load Video/Camera Feed**
   You'll be prompted to select a test video file or use a live webcam feed.

Important Tip:
Install all required Python packages beforehand by running:

pip install -r requirements.txt

This installs dependencies like OpenCV, NumPy, etc.

3. Roles and Functionality

User Role	Features/Responsibilities
System Admin	Manages system settings, retrains ML models, views system logs, and ensures system uptime.
Traffic Officer	Monitors alerts in real-time, responds to incidents, and reviews violation logs for action.
Developer	Maintains and updates code, manages database structure, and retrains or tunes the detection model.

Explanation: Each role has access to specific functionalities, designed to ensure efficiency in system usage and maintenance. Admins focus on management, officers on enforcement, and developers on technical upkeep.

4. Common Problems & Solutions

Problem	Cause	Solution
No image captured	Camera improperly connected	Reconnect the camera and restart the system.
System not detecting violations	Incorrect camera angle or lighting	Adjust the angle and ensure good lighting—natural daylight is preferred.
Buzzer/LED not working	Loose wires	Recheck all wiring connections and verify using Proteus simulation.
Python crash	Missing packages	Install missing libraries using pip install. Check the requirements.txt.
Arduino not responding	COM port not recognized	Check USB cable, reselect the correct COM port in Arduino IDE.

Explanation: These common issues may arise during testing or real-time use. Each entry includes possible root causes and practical solutions.

5. Terminal & System Commands Summary

Task	**Command**
Run Python detection program	python main.py
Install Python packages	pip install -r requirements.txt
Upload program to Arduino	Use Arduino IDE and select the appropriate COM port
Run Proteus simulation	Open .pdsprj file in Proteus 8.17

Explanation: These are the essential commands used during setup and operation. Ensure all prerequisites are met before running commands.

6. Future Deployment Tips (Optional)

- **Cloud Integration:**
  Use platforms like **Firebase** or **ThingsBoard** to push alerts and logs to a cloud dashboard.
- **Edge Computing Upgrade:**
  Replace PC with **Raspberry Pi** to allow the system to work independently on-site.
- **Accelerated ML Processing:**
  Integrate with **NVIDIA Jetson Nano** for faster image processing and better real-time performance.
- **Database Upgrade:**
  Use **PostgreSQL** instead of SQLite for better performance and cloud support.

Explanation: These suggestions help scale the system from a prototype to a production-grade solution.

7. System Testing

Test Type	**Description**
Hardware Test	Use mock objects to simulate a vehicle triggering sensors in reverse direction.
Video Simulation	Run the Python detection script with test videos showing wrong-way scenarios.
Real-time Detection	Deploy system at a road junction and test using actual vehicle movement during daylight hours.
System Response	Check if the buzzer sounds, LED flashes, and image capture is triggered correctly.

Explanation: Proper testing ensures system reliability. Tests can be done at hardware, software, and full-system levels.

8. Last Comments

This manual is a full guide for installing, running, testing, and troubleshooting the **Wrong Way Driving Monitoring System (WWDMS)**.

- Ensure camera positions are optimized and checked weekly.
- Backup data (images, logs) regularly.
- Continue improving the detection model to adapt to new environments.
- Secure configuration and .env files to avoid system compromise.

Explanation: Regular maintenance and proactive improvements are vital for long-term success and reliability.