**Robot Control System Documentation**

**📋 Table of Contents**

1. [System Overview](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#system-overview)
2. [Architecture](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#architecture)
3. [Folder Structure](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#folder-structure)
4. [Installation & Setup](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#installation--setup)
5. [Features](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#features)
6. [API Documentation](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#api-documentation)
7. [Hardware Components](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#hardware-components)
8. [Configuration Files](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#configuration-files)
9. [Future Improvements](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#future-improvements)
10. [Troubleshooting](https://claude.ai/chat/0425a6d8-e0ef-4c45-91ca-83302a535862#troubleshooting)

**System Overview**

WALL-E Robot Control System is a comprehensive robotics platform consisting of:

* **Backend**: Python-based WebSocket server running on Raspberry Pi 5
* **Frontend**: PyQt6 application with real-time control interface (Steam Deck compatible)
* **Camera System**: ESP32-CAM module with proxy server for multi-client streaming
* **Hardware Control**: Dual Pololu Maestro servo controllers, motor drivers, and sensor integration

**Key Technologies**

* **Backend**: Python 3.9.13, asyncio, WebSockets, pygame (audio), GPIO control
* **Frontend**: PyQt6, OpenCV, MediaPipe (gesture detection), pyqtgraph (telemetry)
* **Communication**: WebSocket (control), HTTP (camera streaming), Shared serial management
* **Hardware**: Raspberry Pi 5, ESP32-CAM, Pololu Maestro controllers, Sabertooth motor driver, TB6600 Nema controller, ADC11x5, ACS758x2

**Architecture**

┌──────────────────────────────────────────────────────────────┐

│ Steam Deck / PC │

│ ┌─────────────────────────────────────────────────────┐ │

│ │ PyQt6 Frontend Application │ │

│ │ ├── Home Screen (Emotion/Scene Control) │ │

│ │ ├── Camera Feed (Wave Detection) │ │

│ │ ├── Health Monitor (Telemetry) │ │

│ │ ├── Servo Configuration │ │

│ │ └── Controller Mapping │ │

│ └─────────────────────────────────────────────────────┘ │

└────────────────────┬─────────────────────────────────────────┘

│ WebSocket (ws://10.1.1.230:8766)

│ HTTP (http://10.1.1.230:8081/stream)

┌────────────────────┴─────────────────────────────────────────┐

│ Raspberry Pi 5 │

│ ┌─────────────────────────────────────────────────────┐ │

│ │ Main Backend (main.py) │ │

│ │ ├── WebSocket Server (port 8766) │ │

│ │ ├── Shared Serial Manager │ │

│ │ ├── Scene Engine │ │

│ │ ├── Telemetry System │ │

│ │ └── Audio Controller │ │

│ └─────────────────────────────────────────────────────┘ │

│ ┌─────────────────────────────────────────────────────┐ │

│ │ Camera Proxy (camera\_proxy.py) │ │

│ │ └── MJPEG Rebroadcast Server (port 8081) │ │

│ └─────────────────────────────────────────────────────┘ │

└───────────┬────────────────────┬──────────────┬──────────────┘

│ Serial │ I2C │ Network

┌───────┴────────┐ ┌──────┴──────┐ ┌───┴────┐

│ Maestro 1 & 2 │ │ ADC Sensors │ │ESP32CAM│

│ (Shared Port) │ │ (Current) │ │ WiFi │

└────────────────┘ └─────────────┘ └────────┘

**Folder Structure**

wall-e-robot/

├── 📁 audio/ # Audio files for scenes/emotions

│ ├── track\_001.mp3 # Mapped to emotions/scenes

│ ├── track\_002.wav

│ └── ...

├── 📁 configs/ # Configuration files

│ ├── camera\_config.json # ESP32 camera settings

│ ├── controller\_config.json # Button/control mappings

│ ├── emotion\_buttons.json # Home screen emotions

│ ├── hardware\_config.json # Hardware ports/pins

│ ├── motion\_config.json # Motion control groups

│ ├── scenes.json # Scene definitions

│ ├── servo\_config.json # Servo limits/names

│ └── steamdeck\_config.json # Network/detection settings

├── 📁 icons/ # UI icons and images

│ ├── Home.png, Home\_pressed.png

│ ├── M1.png, M1\_pressed.png

│ ├── Tracking.png, Tracking\_pressed.png

│ └── ...

├── 📁 logs/ # Application logs

│ └── walle\_enhanced\_backend.log

├── 📁 modules/ # Python modules

│ ├── camera\_proxy.py # Camera rebroadcast server

│ └── shared\_serial\_manager.py # Multi-device serial

├── 📁 scenes/ # Scene data (future use)

├── 📄 main.py # Backend entry point

├── 📄 wall\_e\_frontend.py # Frontend application

├── 📄 start\_walle.sh # Startup script

├── 📄 install.sh # Installation script

├── 📄 esp32cam.ino # ESP32-CAM firmware

├── 📄 walle.png # WALL-E image for UI

└── 📄 background.png # UI background

📁 System Files (auto-generated):

├── camera\_proxy.pid # Camera proxy process ID

├── .python-version # pyenv Python version

└── venv/ # Python virtual environment

**Installation & Setup**

**Prerequisites**

* Raspberry Pi 5 with Raspbian OS
* Python 3.9.13 (via pyenv)
* ESP32-CAM module
* Network connection

**Installation Steps**

# 1. Clone repository to Raspberry Pi

git clone <repository> ~/wall-e-robot

cd ~/wall-e-robot

# 2. Run installation script

chmod +x install.sh

./install.sh

# 3. Configure network settings

# Edit configs/steamdeck\_config.json with your IPs

# 4. Flash ESP32-CAM

# Upload esp32cam.ino using Arduino IDE

# 5. Start the system

./start\_walle.sh

# 6. Run frontend on Steam Deck/PC

python wall\_e\_frontend.py

**Features**

**✅ Implemented Features**

**1. Multi-Client Camera Streaming**

* ESP32-CAM with async multi-client support (up to 3 simultaneous)
* Camera proxy server for rebroadcasting
* Real-time camera settings adjustment
* Wave gesture detection using MediaPipe

**2. Servo Control System**

* Dual Maestro controller support (36 channels total)
* Shared serial port management with priority queuing
* Real-time position feedback
* Speed/acceleration control per channel
* Sweep/test functionality
* Channel detection and auto-configuration

**3. Scene & Emotion System**

* Predefined emotional responses
* Audio-synchronized servo movements
* Scene editor with categories
* Home screen quick-access buttons

**4. Health Monitoring**

* Real-time telemetry (CPU, memory, temperature)
* Battery voltage monitoring with alarms
* Dual current sensing (A0, A1)
* Live graphs with data history
* Connection status for all subsystems

**5. Audio System**

* Native Raspberry Pi audio playback
* Multiple format support (MP3, WAV, OGG, M4A)
* Volume control
* Scene-synchronized playback

**6. Safety Features**

* Emergency stop functionality
* Failsafe mode
* GPIO-based limit switches
* Voltage alarms (LOW/CRITICAL)

**7. Network Features**

* SMB file sharing for easy file management
* WebSocket for real-time control
* Multiple client support

**API Documentation**

**WebSocket API (ws://[IP]:8766)**

**Message Format**

All messages use JSON format:

{

"type": "message\_type",

"data": { ... }

}

**Outgoing Commands (Frontend → Backend)**

**Servo Control**

// Set servo position

{

"type": "servo",

"channel": "m1\_ch5", // maestro\_id + channel

"pos": 1500, // 992-2000 typically

"priority": "normal" // emergency|realtime|normal|low|background

}

// Set servo speed

{

"type": "servo\_speed",

"channel": "m1\_ch5",

"speed": 50 // 0-100

}

// Set servo acceleration

{

"type": "servo\_acceleration",

"channel": "m1\_ch5",

"acceleration": 30 // 0-100

}

// Get servo position

{

"type": "get\_servo\_position",

"channel": "m1\_ch5"

}

// Get all positions for a Maestro

{

"type": "get\_all\_servo\_positions",

"maestro": 1 // 1 or 2

}

// Get Maestro info

{

"type": "get\_maestro\_info",

"maestro": 1

}

**Scene Control**

// Play scene/emotion

{

"type": "scene",

"emotion": "happy" // Scene name from scenes.json

}

// Get available scenes

{

"type": "get\_scenes"

}

**Audio Control**

// Play audio

{

"type": "audio",

"command": "play",

"track": "track\_001" // File name without extension

}

// Stop audio

{

"type": "audio",

"command": "stop"

}

// Set volume

{

"type": "audio",

"command": "volume",

"volume": 0.7 // 0.0-1.0

}

**System Control**

// Emergency stop

{

"type": "emergency\_stop"

}

// Failsafe toggle

{

"type": "failsafe",

"state": true

}

// Mode control

{

"type": "mode",

"name": "idle", // idle|demo

"state": true

}

// System status request

{

"type": "system\_status"

}

// Update camera config

{

"type": "update\_camera\_config",

"esp32\_url": "http://10.1.1.203:81/stream"

}

**Gesture Detection**

// Gesture detected (from frontend)

{

"type": "gesture",

"name": "wave"

}

// Toggle tracking

{

"type": "tracking",

"state": true

}

**Incoming Messages (Backend → Frontend)**

**Telemetry Data**

{

"type": "telemetry",

"timestamp": 1234567890.123,

"cpu": 45.2,

"memory": 62.1,

"temperature": 48.5,

"battery\_voltage": 14.8,

"current": 5.2,

"current\_a1": 2.1,

"maestro1": { ... },

"maestro2": { ... },

"audio\_system": {

"connected": true,

"file\_count": 15,

"is\_playing": false,

"current\_track": null,

"volume": 0.7

},

"stream": {

"fps": 30,

"resolution": "640x480",

"latency": 45

},

"system\_status": {

"gpio\_available": true,

"adc\_available": true,

"state": "normal",

"connected\_clients": 2

},

"shared\_managers": { ... }

}

**Servo Responses**

// Single position response

{

"type": "servo\_position",

"channel": "m1\_ch5",

"position": 1500

}

// All positions response

{

"type": "all\_servo\_positions",

"maestro": 1,

"positions": {

"0": 1500,

"1": 1200,

...

},

"total\_channels": 18,

"successful\_reads": 18

}

// Maestro info response

{

"type": "maestro\_info",

"maestro": 1,

"connected": true,

"channels": 18,

"device\_number": 12,

"shared\_port": "/dev/ttyAMA0",

"shared\_manager\_stats": { ... }

}

**Scene Response**

{

"type": "scene\_list",

"scenes": [

{

"label": "happy",

"emoji": "😊",

"category": "Happy",

"duration": 3.0

},

...

]

}

**Camera Proxy HTTP API (http://[IP]:8081)**

**Endpoints**

**Stream Endpoint**

GET /stream

Content-Type: multipart/x-mixed-replace; boundary=frame

**Camera Settings**

GET /camera/settings

Response: {

"xclk\_freq": 10,

"resolution": 5,

"quality": 12,

"brightness": 0,

"contrast": 0,

"saturation": 0,

"h\_mirror": false,

"v\_flip": false

}

POST /camera/settings

Body: { settings object }

POST /camera/setting/{setting\_name}

Params: value=<value>

**Status**

GET /stats

Response: {

"fps": 30,

"frame\_count": 1234,

"dropped\_frames": 2,

"latency": 45,

"has\_frame": true,

"stream\_url": "http://10.1.1.203:81/stream",

"connected\_to\_esp32": true,

"connection\_errors": 0,

"uptime": 3600,

"status": "connected",

"esp32\_settings": { ... }

}

**Hardware Components**

**Core Components**

* **Raspberry Pi 5**: Main controller
* **ESP32-CAM**: Video streaming
* **Pololu Maestro 18-channel (x2)**: Servo control (Device #12, #13)
* **Sabertooth 2x60**: Motor driver
* **TB6600**: Stepper motor driver (future)
* **ADS1115**: ADC for current sensing

**Pin Assignments**

**GPIO Pins (BCM)**

* **16**: Motor step pin
* **12**: Motor direction pin
* **13**: Motor enable pin
* **26**: Limit switch pin
* **25**: Emergency stop pin

**Serial Ports**

* **/dev/ttyAMA0**: Shared by both Maestro controllers (9600 baud)
* **/dev/ttyAMA1**: Sabertooth motor driver (9600 baud)

**I2C**

* **SDA/SCL**: ADS1115 ADC for battery/current monitoring

**Configuration Files**

**hardware\_config.json**

{

"hardware": {

"maestro1": {

"port": "/dev/ttyAMA0",

"baud\_rate": 9600,

"device\_number": 12

},

"maestro2": {

"port": "/dev/ttyAMA0",

"baud\_rate": 9600,

"device\_number": 13

},

"sabertooth": {

"port": "/dev/ttyAMA1",

"baud\_rate": 9600

},

"gpio": { ... },

"timing": { ... },

"audio": { ... }

}

}

**scenes.json**

{

"happy": {

"emoji": "😊",

"category": "Happy",

"duration": 3.0,

"audio\_file": "track\_002",

"servos": {

"m1\_ch0": {"target": 1500, "speed": 50},

"m1\_ch1": {"target": 1200, "speed": 30}

}

}

}

**servo\_config.json**

{

"m1\_ch0": {

"name": "Head Tilt",

"min": 992,

"max": 2000,

"speed": 50,

"accel": 30

}

}

**Future Improvements**

**🎯 Priority 1 - Immediate Enhancements**

**1. Servo Config Checkbox Feature**

Add selective live updates to servo configuration screen:

# In ServoConfigScreen class

- Add checkbox widget per channel

- Only send updates when checked AND slider moved

- Batch update option for multiple servos

- "Enable All" / "Disable All" buttons

**2. Bluetooth Steam Deck Controller**

Implement direct Bluetooth gamepad support:

# New bluetooth\_controller.py module

- Steam Deck controller pairing

- Button/axis mapping to robot functions

- Haptic feedback support

- Low-latency command transmission

- Connection status indicator in UI

**3. Enhanced Controller Mapping**

Improve the controller configuration system:

# Enhancements needed:

- Visual controller layout editor

- Macro recording for complex movements

- Sensitivity curves for analog inputs

- Dead zone configuration

- Profile save/load system

- Test mode with visual feedback

**🎯 Priority 2 - System Improvements**

**4. Motor Driver Integration**

Complete TB6600 and Sabertooth control:

# Motor control features:

- Stepper motor positioning system

- Closed-loop control with encoders

- Acceleration/deceleration profiles

- Coordinated multi-axis movement

- Homing sequences

- Speed ramping for Sabertooth

**5. Scene System Enhancements**

Expand scene capabilities:

# Scene improvements:

- Timeline editor for complex sequences

- Conditional scene triggers

- Scene chaining and transitions

- Loop/repeat options

- Variable speed playback

- Import/export scene files

**🎯 Priority 3 - Advanced Features**

**🔧 Code Quality Improvements**

1. **Error Recovery**
   * Automatic reconnection for all components
   * Graceful degradation when hardware unavailable
   * Error queue with retry logic
   * State persistence across restarts
2. **Performance Optimization**
   * Implement connection pooling
   * Add caching layer for frequent requests
   * Optimize frame processing pipeline
   * Reduce memory allocations in hot paths

**📊 Proposed Development Roadmap**

**Phase 1 (Current Sprint)**

* ✅ Servo config checkbox feature
* ⬜ Bluetooth controller initial support
* ⬜ Enhanced controller mapping UI

**Phase 2 (Next Month)**

* ⬜ TB6600 stepper integration
* ⬜ Sabertooth advanced control
* ⬜ Scene timeline editor

**Phase 3 (Q2 2025)**

* ⬜ Web interface development
* ⬜ AI behavior system
* ⬜ Advanced telemetry

**Phase 4 (Q3 2025)**

* ⬜ Multi-robot support
* ⬜ Simulation environment
* ⬜ Cloud services integration

**Troubleshooting**

**Common Issues**

**Camera Not Connecting**

# Check ESP32 is on network

ping 10.1.1.203

# Restart camera proxy

./manage\_smb.sh restart

# Check camera proxy logs

tail -f logs/walle\_enhanced\_backend.log

**Servo Not Responding**

# Check serial port

ls -la /dev/ttyAMA\*

# Test Maestro connection

screen /dev/ttyAMA0 9600

# Check device numbers in config

cat configs/hardware\_config.json

**WebSocket Connection Failed**

# Check backend is running

ps aux | grep main.py

# Check port is open

netstat -tulpn | grep 8766

# Test WebSocket

wscat -c ws://10.1.1.230:8766

**Low Battery Warning**

* Battery < 13.2V: LOW warning
* Battery < 11.0V: CRITICAL - land immediately
* Check voltage divider calibration in telemetry system

**Debug Commands**

# View real-time logs

tail -f logs/walle\_enhanced\_backend.log

# Check system resources

htop

# Monitor I2C devices

i2cdetect -y 1

# Test audio

speaker-test -t wav

# Check network

ip addr show