CR5 Automatic Launcher

Operating System Technical Documentation and Network Configuration

Robotics team (that would be nice... but that's just me)

October 17, 2025

Contents

1 Detailed Script Operation Guide

This section provides a comprehensive explanation of how the automated startup script works, including detailed analysis of each step and complete usage instructions.

1.1 Script Architecture Overview

The `dobot_startup.sh` script follows a modular architecture with clear separation of concerns:

- **Configuration Management**: Centralized variable definitions
- **Logging System**: Color-coded output with different log levels
- **Error Handling**: Robust error checking with graceful failure handling
- **Process Management**: PID tracking and cleanup mechanisms
- **Interactive Mode**: User guidance and confirmations

1.2 Step-by-Step Process Analysis

1.2.1 Step 1: Network Interface Detection and Configuration

Function: `detect_network_interface()` **Purpose:** Automatically detects available network interfaces and allows user selection.

Process Flow:

1. **Interface Discovery:**

```
interfaces=$(ip link show | grep -E "^[0-9]+:" | grep -v lo | awk -F': ' '{print $2}' | cut -d'@' -f1)
```

2. **User Selection:** Presents interactive menu with all available interfaces
3. **Manual Override:** Allows manual interface name input if needed
4. **Validation:** Stores selected interface in `NETWORK_INTERFACE` variable

Error Conditions:

- No network interfaces found (excluding loopback)
- Invalid user selection
- Manually entered interface doesn't exist

Function: `configure_network()` **Purpose:** Configures the selected network interface with static IP for robot communication.

Technical Details:

- **Target IP:** 192.168.5.100/24 (configurable)
- **Robot IP:** 192.168.5.1 (standard Dobot configuration)
- **Subnet:** 192.168.5.0/24 (Class C private network)

Process Flow:

1. **Interface Existence Check:**

```
if ! ip link show "$NETWORK_INTERFACE" &>/dev/null; then
log_error "Network␣interface␣$NETWORK_INTERFACE␣does␣not␣exist!"
return 1
fi
```

2. **NetworkManager Conflict Prevention:**

```
# Disable NetworkManager management to avoid conflicts
if command -v nmcli &>/dev/null; then
sudo nmcli dev set "$NETWORK_INTERFACE" managed no
sudo systemctl restart NetworkManager
sleep 2  # Wait for NetworkManager to restart
fi
```

3. **IP Conflict Detection:**

```
if ip addr show "$NETWORK_INTERFACE" | grep -q "$LOCAL_IP"; then
log_warning "IP␣$LOCAL_IP␣already␣configured"
return 0
fi
```

4. **IP Address Assignment:**

```
sudo ip addr add "$LOCAL_IP/24" dev "$NETWORK_INTERFACE"
```

5. **Interface Activation:**

```
sudo ip link set "$NETWORK_INTERFACE" up
```

NetworkManager Integration:

- **Automatic Detection:** Checks if NetworkManager is installed using `command -v nmcli`

- **Management Disable:** Prevents NetworkManager from automatically managing the robot interface
- **Service Restart:** Ensures NetworkManager applies the new configuration
- **Graceful Fallback:** Continues operation even if NetworkManager commands fail
- **Conflict Prevention:** Eliminates interference between manual IP configuration and automatic DHCP

1.2.2 Step 2: Robot Connectivity Verification

Function: `test_robot_connectivity()` **Purpose:** Verifies network connectivity to the robot before proceeding with software initialization.

Technical Implementation:

```
1    if ping -c 3 -W 2 "$ROBOT_IP" &>/dev/null; then
2    log_success "Robot is reachable at $ROBOT_IP"
3    return 0
4    else
5    log_error "Cannot reach robot at $ROBOT_IP"
6    return 1
7    fi
```

Parameters Explained:

- `-c 3`: Send exactly 3 ping packets
- `-W 2`: Wait maximum 2 seconds for response
- `&>/dev/null`: Suppress ping output (only check return code)

Failure Diagnostics: The script provides comprehensive troubleshooting guidance:

- Robot power status check
- IP configuration verification
- Physical connection validation
- Network configuration review

1.2.3 Step 3: ROS2 Environment Validation

Function: `check_ros2_environment()` **Purpose:** Ensures ROS2 environment is properly configured and workspace is available.

Environment Checks:

1. **ROS2 Distribution Check:**

```
1        if [ -z "$ROS_DISTRO" ]; then
2        log_error "ROS2 environment not sourced!"
3        return 1
4        fi
```

2. **Workspace Discovery:**

```
if [ -f "$HOME/dobot_v3/install/setup.bash" ]; then
source "$HOME/dobot_v3/install/setup.bash"
log_success "Workspace␣sourced␣successfully"
fi
```

Required Environment Variables:

- `ROS_DISTRO`: Must be set (typically "humble")
- `ROS_VERSION`: Should be "2" for ROS2
- `AMENT_PREFIX_PATH`: Workspace package paths

1.2.4 Step 4: Robot Driver Initialization

Function: `start_robot_drivers()` **Purpose:** Launches the low-level robot drivers that communicate directly with the Dobot CR5 hardware.

Process Implementation:

1. **Duplicate Process Check:**

```
if pgrep -f "dobot_bringup_ros2.launch.py" &>/dev/null;
    then
log_warning "Robot␣drivers␣appear␣to␣be␣already␣running
   "
return 0
fi
```

2. **Driver Launch:**

```
ros2 launch dobot_bringup_v3 dobot_bringup_ros2.launch.
   py &
DRIVER_PID=$!
```

3. **Initialization Wait:**

```
sleep 30  # Allow 30 seconds for driver initialization
```

4. **Process Validation:**

```
if kill -0 $DRIVER_PID 2>/dev/null; then
echo $DRIVER_PID > /tmp/dobot_driver.pid
return 0
fi
```

Driver Responsibilities:

- Hardware communication protocol handling

- Joint state publishing (`/joint_states` topic)
- Command interface for robot control
- Safety monitoring and emergency stops
- Robot status reporting

1.2.5 Step 5: MoveIt Planning System Launch

Function: `start_moveit_system()` **Purpose:** Initializes the MoveIt motion planning framework for high-level robot control.

System Components:

- **Move Group Node**: Core planning and execution coordinator
- **Planning Scene**: 3D environment representation with obstacles
- **Planning Pipeline**: Motion planning algorithms (OMPL)
- **Controller Manager**: Interface to robot controllers
- **Reworked_map**: Activation of reworked_map_node to translate zed messages into the desired format
- **RViz Integration**: Visualization and debugging interface

Launch Process:

1. **Conflict Detection:**

```
    if pgrep -f "full_bringup.launch.py" &>/dev/null; then
    log_warning "MoveIt system appears to be already
       running"
    return 0
    fi

    if pgrep -f "reworked_map_node" &>/dev/null; then
    log_warning "reworked_map_node appears to be already
       running"
    return 0
    fi
```

2. **System Launch:**

```
    ros2 launch cr5_moveit full_bringup.launch.py &
    MOVEIT_PID=$!

    ros2 run cr5_moveit_cpp_demo reworked_map_node &
    REWORKED_MAP_PID=$!
```

3. **Extended Initialization:**

```
    sleep 30  # MoveIt requires longer initialization time
```

1.2.6 Step 6: System Readiness Verification

Function: `verify_system_readiness()` **Purpose:** Performs comprehensive system checks to ensure all components are operational.

Verification Tests:

1. **Robot Topics Check:**

```
timeout 10 ros2 topic list | grep -q "/joint_states" || {
  log_error "Robot joint_states topic not found!"
  return 1
}
```

2. **Planning Service Check:**

```
timeout 10 ros2 service list | grep -q "plan_kinematic_path" || {
  log_error "MoveIt planning service not found!"
  return 1
}
```

Critical System Topics:

- `/joint_states`: Real-time joint position feedback
- `/robot_description`: URDF model information
- `/planning_scene`: 3D environment representation
- `/move_group/display_planned_path`: Trajectory visualization

Essential Services:

- `/plan_kinematic_path`: Motion planning service
- `/execute_trajectory`: Trajectory execution service
- `/get_planning_scene`: Environment query service
- `/clear_octomap`: Obstacle map management

1.2.7 Step 7: Movement Node Launch Options

Function: `start_movement_node()` **Purpose:** Provides user options for launching the actual movement control application.

Available Options:

1. **Modular Implementation:**

```
ros2 run cr5_moveit_cpp_demo move_cr5_node_modular
```

2. **Original Monolithic Version:**

```
ros2 run cr5_moveit_cpp_demo move_cr5_node
```

Interactive Launch:

```
read -p "Start movement node now? (y/N): " -n 1 -r
if [[ $REPLY =~ ^[Yy]$ ]]; then
ros2 run cr5_moveit_cpp_demo move_cr5_node_modular
fi
```

1.3 Process Management and Cleanup

1.3.1 PID Tracking System

Purpose: Maintains process identifiers for proper cleanup and monitoring.

Implementation:

- **Driver PID Storage:** `/tmp/dobot_driver.pid`
- **MoveIt PID Storage:** `/tmp/dobot_moveit.pid`
- **Process Validation:** `kill -0 $PID` for existence check

1.3.2 Cleanup Function

Function: `cleanup()` **Purpose:** Ensures proper termination of all spawned processes on script exit.

Cleanup Process:

1. Read stored PIDs from temporary files
2. Validate process existence with `kill -0`
3. Send termination signals to active processes
4. Remove temporary PID files
5. Log cleanup operations

Trap Implementation:

```
trap cleanup EXIT
```

This ensures cleanup occurs on:

- Normal script completion
- User interruption (Ctrl+C)
- Script errors and early termination
- System signals (SIGTERM, SIGHUP)

2 Complete Usage Instructions

2.1 Prerequisites

Before using the automated startup script, ensure the following prerequisites are met:

2.1.1 System Requirements

- **Operating System:** Ubuntu 22.04 LTS (recommended)
- **ROS2 Distribution:** ROS2 Humble Hawksbill
- **Network Interface:** Ethernet port for robot connection
- **Sudo Access:** Required for network configuration
- **Workspace:** Properly built ROS2 workspace with Dobot packages

2.1.2 Software Dependencies

```
# Core ROS2 packages
sudo apt update
sudo apt install ros-humble-desktop-full

# MoveIt2 packages
sudo apt install ros-humble-moveit

# Additional utilities
sudo apt install net-tools iputils-ping
```

Listing 1: Install Required Packages

2.1.3 Workspace Preparation

```
# Navigate to workspace
cd ~/dobot_v3

# Build the workspace
colcon build --packages-select cr5_moveit_cpp_demo

# Source the workspace
source install/setup.bash
```

Listing 2: Workspace Setup

2.2 Initial Setup Process

2.2.1 Script Preparation

1. **Navigate to Package Directory:**

```
cd ~/dobot_v3/src/cr5_moveit_cpp_demo
```

2. **Verify Script Permissions:**

```
ls -la dobot_startup.sh dobot_stop.sh
```

3. **Make Scripts Executable (if needed):**

```
chmod +x dobot_startup.sh dobot_stop.sh
```

2.2.2 Robot Hardware Setup

1. **Physical Connections:**
   - Connect Ethernet cable between PC and robot
   - Ensure robot is powered on
   - Verify robot's teach pendant shows network connectivity
2. **Robot IP Configuration:**
   - Access robot's network settings via teach pendant
   - Set robot IP to `192.168.5.1`
   - Set subnet mask to `255.255.255.0`
   - Save configuration and restart robot if required

2.3 Step-by-Step Execution Guide

2.3.1 Method 1: Fully Automated Execution

Single Command Launch:

```
cd ~/dobot_v3/src/cr5_moveit_cpp_demo
./dobot_startup.sh
```

Expected Interaction Flow:

1. **Interface Selection:**

```
Available network interfaces:
1) enp2s0
2) wlp3s0
3) Manual Input
Please select: 1
```

2. **Network Configuration:**

```
[INFO] Configuring network interface enp2s0...
[SUCCESS] IP address configured successfully
```

3. **Robot Connectivity Test:**

```
[INFO] Testing connectivity to robot at 192.168.5.1...
[SUCCESS] Robot is reachable at 192.168.5.1
```

4. **System Launch Sequence:**

```
[INFO] Starting Dobot CR5 drivers...
[INFO] Waiting for drivers to initialize (30 seconds)...
[SUCCESS] Robot drivers started successfully

[INFO] Starting MoveIt and full robot system...
[INFO] Waiting for MoveIt to initialize (30 seconds)...
[SUCCESS] MoveIt system started successfully
```

5. **Final Confirmation:**

```
[SUCCESS] DOBOT CR5 SYSTEM READY!
Start movement node now? (y/N): y
```

2.3.2 Method 2: Manual Step Verification

For debugging or learning purposes, you can manually execute individual steps:

1. **Network Configuration:**

```
# Replace enp2s0 with your interface name
sudo ip addr add 192.168.5.100/24 dev enp2s0
sudo ip link set enp2s0 up

# Test connectivity
ping -c 3 192.168.5.1
```

2. **ROS2 Environment:**

```
# Source ROS2
source /opt/ros/humble/setup.bash

# Source workspace
source ~/dobot_v3/install/setup.bash
```

```
6
7       # Verify environment
8       echo $ROS_DISTRO
```

3. **Robot Drivers:**

```
1       # Terminal 1: Start drivers
2       ros2 launch dobot_bringup_v3 dobot_bringup_ros2.launch.
           py
```

4. **MoveIt System:**

```
1       # Terminal 2: Start MoveIt (after drivers are running)
2       ros2 launch cr5_moveit full_bringup.launch.py
```

5. **Movement Node:**

```
1       # Terminal 3: Start movement application
2       ros2 run cr5_moveit_cpp_demo move_cr5_node_modular
```

2.4 System Shutdown and Cleanup

ATTENTION: Use this process ONLY if the drivers were activated manually.

2.4.1 Automated Cleanup

```
1     cd ~/dobot_v3/src/cr5_moveit_cpp_demo
2     ./dobot_stop.sh
```

Cleanup Output:

```
[INFO] Stopping movement nodes...
[SUCCESS] Movement nodes stopped

[INFO] Stopping MoveIt system...
[SUCCESS] MoveIt system stopped

[INFO] Stopping robot drivers...
[SUCCESS] Robot drivers stopped

[SUCCESS] All Dobot processes stopped successfully
```

2.4.2 Manual Cleanup (if needed)

```
# Kill all Dobot-related processes
pkill -f dobot
pkill -f moveit
pkill -f move_cr5

# Remove network configuration (optional)
sudo ip addr del 192.168.5.100/24 dev enp2s0
```

2.5 Troubleshooting Common Issues

2.5.1 Network Configuration Problems

Issue: Permission denied when configuring network

```
# Solution: Ensure sudo access
sudo -v  # Verify sudo access
./dobot_startup.sh  # Try again
```

Issue: Network interface not found

```
# List all interfaces
ip link show

# Use correct interface name in script or select manually
```

2.5.2 Robot Connectivity Issues

Issue: Robot not responding to ping

- Verify robot power and network LED status
- Check Ethernet cable connection
- Confirm robot IP configuration (192.168.5.1)
- Test with different Ethernet port if available

Issue: Intermittent connectivity

```
# Disable NetworkManager for the interface
sudo nmcli dev set enp2s0 managed no
sudo systemctl restart NetworkManager

# Reconfigure network manually
sudo ip addr add 192.168.5.100/24 dev enp2s0
```

2.5.3 ROS2 Environment Issues

Issue: ROS_DISTRO not set

```
# Source ROS2 environment
source /opt/ros/humble/setup.bash

# Add to bashrc for persistence
echo "source␣/opt/ros/humble/setup.bash" >> ~/.bashrc
```

Issue: Workspace not found

```
# Rebuild workspace
cd ~/dobot_v3
colcon build --packages-select cr5_moveit_cpp_demo

# Source workspace
source install/setup.bash
```

2.5.4 Driver and MoveIt Issues

Issue: Drivers fail to start

- Check robot connectivity first
- Verify no conflicting processes are running
- Review driver logs for specific error messages
- Ensure all required packages are installed

Issue: MoveIt initialization fails

```
# Check for missing dependencies
ros2 pkg list | grep moveit

# Verify robot description
ros2 param get /robot_description robot_description
```

2.6 Advanced Configuration Options

2.6.1 Customizing Network Settings

Edit the script variables at the top of `dobot_startup.sh`:

```
# Custom robot IP
ROBOT_IP="192.168.1.100"

# Custom local IP
LOCAL_IP="192.168.1.50"

# Custom interface (bypass detection)
NETWORK_INTERFACE="eth0"
```

2.6.2 Timeout Adjustments

Modify initialization wait times for slower systems:

```
    # In start_robot_drivers()
    sleep 45  # Increase from 30 seconds

    # In start_moveit_system()
    sleep 60  # Increase from 45 seconds
```

2.6.3 Debug Mode

Enable verbose output for troubleshooting:

```
    # Add at beginning of script
    set -x  # Enable debug output
    set -v  # Enable verbose mode
```