Backend Developer Instructions

Mode Selector Feature Implementation

GRP Pipe Ring Stiffness Test Machine | January 2026

1. Overview

We need to add support for Remote/Local mode selection. This involves reading and writing a new boolean variable from the PLC, and exposing it through the API.

2. New PLC Variable

The PLC has a new variable for mode selection:

Address	Name	Type	Access	Values
DB3.DBX25.0	Remote_Mode	Bool	Read/Write	0=Local, 1=Remote

Snap7 Access:

```
# Read Remote_Mode
remote_mode = plc.read_db_bool(3, 25, 0)  # DB3, Byte 0, Bit 7

# Write Remote_Mode
plc.write_db_bool(3, 25, 0, True)   # Set to Remote
plc.write_db_bool(3, 25, 0, False)  # Set to Local
```

3. Files to Modify

File	Changes Required
plc_connector.py	No changes (read_db_bool/write_db_bool already exist)
data_service.py	Add remote_mode to get_live_data() return
command_service.py	Add set_remote_mode() and get_remote_mode() methods
routes/api.py	Add /api/mode endpoints
models/schemas.py	Add remote_mode to LiveData schema (if using Pydantic)

4. data_service.py Changes

Add remote_mode to the get_live_data() return dictionary:

```
def get_live_data(self) -> dict:
    return {
        # ... existing fields ...
        'servo_ready': self.plc.read_input_bit(0, 0),
        'servo_error': self.plc.read_input_bit(0, 1),
        # ... more fields ...

        # ═══ NEW: Add this field ═══
        'remote_mode': self.plc.read_db_bool(3, 25, 0),  # DB3.DBX25.0
    }
```

5. command_service.py Changes

Add these two methods to the CommandService class:

```
# ═══════════════════════════════════════════════════════════
# MODE CONTROL - NEW METHODS
# ═══════════════════════════════════════════════════════════

def set_remote_mode(self, is_remote: bool):
    """
    Set control mode
    is_remote=True  -> Remote mode (Web interface)
    is_remote=False -> Local mode (Physical buttons)
    """
    self.plc.write_db_bool(3, 25, 0, is_remote)

def get_remote_mode(self) -> bool:
    """Get current control mode"""
    return self.plc.read_db_bool(3, 25, 0)
```

6. New API Endpoints

Add these endpoints to routes/api.py:

Method	Endpoint	Description
GET	/api/mode	Get current mode
POST	/api/mode/local	Switch to Local mode
POST	/api/mode/remote	Switch to Remote mode

FastAPI Implementation:

```
# ═══════════════════════════════════════════════════════════
# MODE CONTROL ENDPOINTS
# ═══════════════════════════════════════════════════════════

@router.get("/mode")
async def get_mode():
    """Get current control mode"""
    remote_mode = command_service.get_remote_mode()
    return {
        "remote_mode": remote_mode,
        "mode": "remote" if remote_mode else "local"
    }

@router.post("/mode/local")
async def set_local_mode():
    """Switch to Local mode (Physical buttons)"""
    command_service.set_remote_mode(False)
    return {"success": True, "mode": "local"}

@router.post("/mode/remote")
async def set_remote_mode():
    """Switch to Remote mode (Web interface)"""
    command_service.set_remote_mode(True)
    return {"success": True, "mode": "remote"}
```

7. WebSocket live_data Update

The remote_mode will automatically be included in live_data broadcasts since we added it to get_live_data(). No additional WebSocket changes needed.

```
# live_data broadcast will now include:
{
    "actual_force": 12.5,
    "servo_ready": true,
    // ... other fields ...
    "remote_mode": true    // ← NEW FIELD
}
```

8. Response Schemas (if using Pydantic)

```
# schemas.py

class LiveData(BaseModel):
    # ... existing fields ...
    servo_ready: bool
    servo_error: bool
    remote_mode: bool = False  # ← ADD THIS

class ModeResponse(BaseModel):
    remote_mode: bool
    mode: str  # "local" or "remote"
```

9. Testing Checklist

☐ GET /api/mode returns current mode
☐ POST /api/mode/local sets remote_mode to false
☐ POST /api/mode/remote sets remote_mode to true
☐ live_data WebSocket includes remote_mode field
☐ Mode persists in PLC after API restart
☐ Read/Write to DB3.DBX25.0 works correctly

10. Quick Reference

Item	Value
PLC Address	DB3.DBX25.0 (DB=3, Byte=0, Bit=7)
Read Method	plc.read_db_bool(3, 25, 0)
Write Method	plc.write_db_bool(3, 25, 0, value)
False (0)	LOCAL mode - Physical buttons active
True (1)	REMOTE mode - Web interface active

Document Version: 1.0 | Created: January 2026