**Mini Project M2M Raspberry PI as PLC for DoBot**

**Students: Bâtcă George Leonard, Pătrașcu Matei**

Objective:  
Use a Raspberry Pi as a PLC to control two Dobot robot arms via HTTP requests, monitor the system with a camera, and provide a web-based interface for control and live streaming. Integrate basic OpenCV functions for simple visual feedback.

## 1. System Architecture

* **Raspberry Pi** (acts as PLC/controller)
* **2 Dobot Arms** (controlled via HTTP requests or Python API)
* **USB or Pi Camera** (for live monitoring and basic vision tasks)
* **Web-based Interface** (for sending commands and viewing the camera stream)

## 2. Project Phases

### Phase 1: Setup & Initial Configuration

* Install Raspberry Pi OS and connect to your LAN (Wi-Fi or Ethernet)[1](https://www.raspberrypi.com/tutorials/how-to-use-a-raspberry-pi-in-kiosk-mode/).
* Set up Python environment and install required libraries:
  + Flask or FastAPI (for HTTP server/web interface)
  + OpenCV (for camera access and simple vision)
  + Dobot API or HTTP client for robot arm control

### Phase 2: Dobot Control Implementation

* Connect both Dobot arms to the Raspberry Pi via USB.
* Use the Dobot Python SDK or HTTP interface to send movement commands[2](https://www.instructables.com/Raspberry-Pi-and-Wiimote-controlled-Robot-Arm/).
* Create Python functions to control each arm (move, pick, place, etc.).

### Phase 3: Web Interface & Button Integration

* Build a simple web page (using Flask/FastAPI) that:
  + Shows live camera feed (MJPEG or similar)
  + Has buttons for sending HTTP POST requests to the Dobot arms for predefined actions (e.g., Move Arm 1 Home, Pick with Arm 2)
* Each button triggers a Python function that sends the correct command to the appropriate Dobot.

### Phase 4: Camera Monitoring & OpenCV Integration

* Use OpenCV to capture and stream video from the camera.
* Optionally, implement basic visual recognition (e.g., color detection or motion detection) for monitoring, but keep it simple since QR codes are not required[3](https://www.hiwonder.com/products/armpi-fpv).

### Phase 5: Testing & Documentation

* Test all controls and the live stream.
* Document the system architecture, code, and usage.
* Prepare a design document and schedule a sprint review.

## 3. Deliverables

* **Design Document**: System overview, architecture diagram, and component descriptions.
* **Sprint Review**: Demonstration of the working system (web interface, Dobot control, camera stream).
* **Source Code**: Python scripts for server, Dobot control, and camera streaming.
* **User Guide**: Instructions for setup, operation, and troubleshooting.

## 4. Notes and Simplifications

* **No tactile buttons:** All controls will be via the web interface (on any device in your LAN).
* **No QR code/advanced vision:** Use basic OpenCV features (like color or motion detection) for demonstration if needed.
* **No real-time joint angle displays:** Focus on sending commands and confirming movement via the camera.
* **No special hardware requirements:** Use standard USB power for the Raspberry Pi[4](https://thepihut.com/blogs/raspberry-pi-tutorials/how-do-i-power-my-raspberry-pi). Industrial-grade GPIO and surge protection are not necessary for a prototype.
* **No external buttons:** All user input is through the web interface.

## 5. Example Web Interface Workflow

1. User opens the web page on a computer or phone in the LAN.
2. User sees the live camera stream.
3. User clicks a button (e.g., "Move Arm 1 Home").
4. The Raspberry Pi receives the HTTP POST, sends the command to the Dobot, and the movement is visible on the stream.

This plan is streamlined for your available hardware and skills, ensuring a functional and presentable M2M mini project without unnecessary complexity.