**Design Document: Raspberry Pi-Based M2M Control System for Dual Dobot Arms with Camera Monitoring**

**1. Introduction**

This document details the design of a machine-to-machine (M2M) automation system using a Raspberry Pi as a PLC to control two Dobot robotic arms, a conveyor belt, and a camera for system monitoring. The system integrates Dockerized services, Node-RED, MQTT communication, and Python code to enable flexible, remote, and automated control in a factory scenario.

**2. System Architecture**

**2.1 Overview**

The system architecture is visualized in the provided diagram[[1]](#fn1). Key components include:

* **Raspberry Pi**: Central controller running Docker containers for Python code, Node-RED, and an MQTT broker. Provides USB connectivity to both Dobot arms and the webcam.
* **Dobot Arms**: Two robot arms connected via USB, performing pick-and-place operations on a conveyor belt.
* **Webcam**: Connected to the Raspberry Pi for real-time monitoring and potential computer vision tasks.
* **Conveyor Belt**: Moves items between the Dobot arms, equipped with a sensor for tag detection.
* **Factory Node-RED**: External Node-RED instance in the factory, communicating with the Raspberry Pi Node-RED via MQTT.
* **VPN**: Secures remote access for developers.
* **Developer/Router**: Enables remote development and monitoring over WiFi.

**2.2 Component Details**

|  |  |
| --- | --- |
| Component | Description |
| Raspberry Pi | Runs Docker, hosts Python code, Node-RED, MQTT broker, and manages USB devices. |
| Docker | Containerizes services for modularity and ease of deployment. |
| Node-RED | Visual programming for event-driven logic and MQTT messaging. |
| MQTT Broker | Handles message passing between factory and Raspberry Pi. |
| Dobot Arms | Execute pick-and-place operations; controlled via Python scripts over USB. |
| Conveyor Belt | Moves items; starts/stops based on sensor input and robot commands. |
| Sensor | Detects presence of NFC tag on conveyor, triggers belt stop. |
| Webcam | Provides live video stream for monitoring and potential OpenCV-based processing. |
| VPN | Allows secure, remote developer access to the system. |

**3. Data and Control Flow**

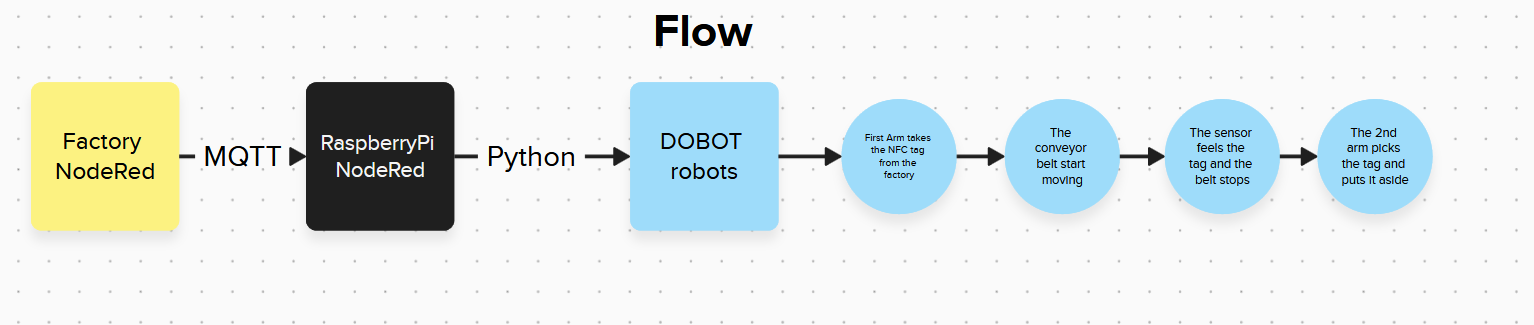
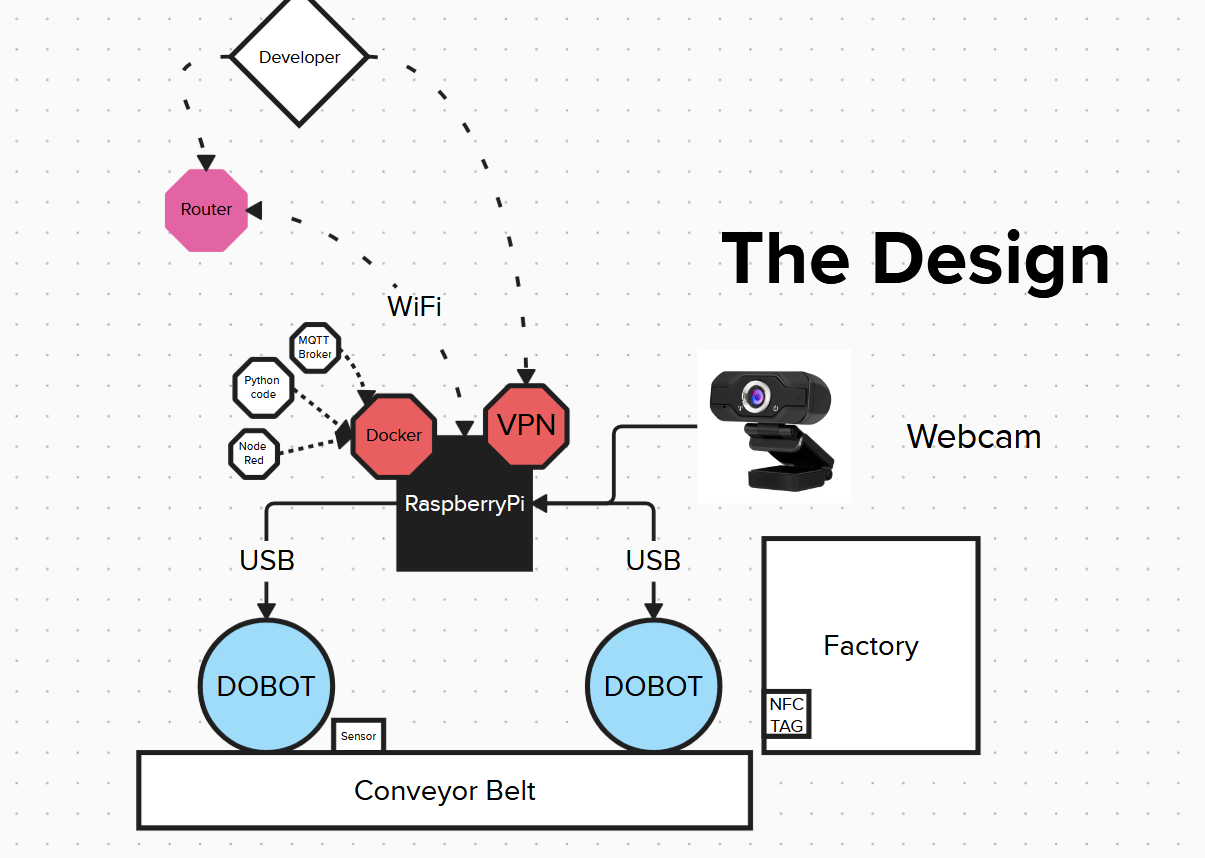
**3.1 Flow Description**

Based on the flow diagram[[2]](#fn2):

1. **Factory Node-RED** sends an MQTT message to the Raspberry Pi Node-RED instance.
2. **Raspberry Pi Node-RED** processes the message and triggers Python code.
3. **Python Code** on the Raspberry Pi controls the Dobot robots:
   * The first Dobot arm takes the NFC tag from the factory.
   * The conveyor belt starts moving.
   * When the sensor detects the tag, the conveyor stops.
   * The second Dobot arm picks the tag and puts it aside.
4. **Webcam** streams the process for monitoring and can be used for simple computer vision tasks if needed.
5. **Developer** can access the system remotely via VPN for monitoring, updates, or troubleshooting.

**3.2 Sequence Diagram**

sequenceDiagram  
 participant Factory\_NodeRED  
 participant MQTT  
 participant RPi\_NodeRED  
 participant Python  
 participant Dobot1  
 participant Conveyor  
 participant Sensor  
 participant Dobot2  
 participant Webcam  
  
 Factory\_NodeRED->>MQTT: Send tag handling command  
 MQTT->>RPi\_NodeRED: Receive message  
 RPi\_NodeRED->>Python: Trigger Dobot1 action  
 Python->>Dobot1: Pick NFC tag  
 Dobot1->>Conveyor: Place tag  
 RPi\_NodeRED->>Conveyor: Start belt  
 Sensor->>RPi\_NodeRED: Tag detected  
 RPi\_NodeRED->>Conveyor: Stop belt  
 RPi\_NodeRED->>Python: Trigger Dobot2 action  
 Python->>Dobot2: Pick tag and set aside  
 Webcam-->>Developer: Stream live video



**4. Software Components**

**4.1 Dockerized Services**

* **Python Service**: Controls Dobot arms using [DobotPi][[3]](#fn3) or Dobot Python SDK.
* **Node-RED**: Orchestrates logic and integrates MQTT messages.
* **MQTT Broker**: Handles publish/subscribe messaging between factory and Raspberry Pi.

**4.2 Dobot Control**

* Python scripts issue commands to Dobot arms for precise pick-and-place tasks.
* Error handling and status feedback are implemented for reliability[[3]](#fn3).

**4.3 Camera Integration**

* Webcam is accessed via Python (OpenCV) for live streaming and potential object detection.
* Video stream can be embedded in a web dashboard for monitoring[[4]](#fn4)[[5]](#fn5)[[6]](#fn6).

**5. Hardware Integration**

* **USB Connections**: Both Dobot arms and the webcam are connected to the Raspberry Pi via USB.
* **Sensor**: Positioned on the conveyor belt to detect the presence of the NFC tag and trigger logic in Node-RED.
* **Conveyor Belt**: Electrically controlled by the Raspberry Pi (via GPIO or relay module, if needed).

**6. Security and Remote Access**

* **VPN**: Ensures secure remote access for developers to monitor, update, or troubleshoot the system.
* **WiFi**: Connects Raspberry Pi to the factory network and enables communication with external systems.

**7. System Operation Example**

1. Factory Node-RED initiates a process by sending an MQTT message.
2. Raspberry Pi Node-RED receives the message and triggers the first Dobot to pick the NFC tag.
3. The conveyor belt moves the tag until the sensor detects it.
4. The belt stops, and the second Dobot picks up the tag and places it aside.
5. The webcam streams the entire process for monitoring and possible vision-based validation.

**8. Extensibility**

* **Computer Vision**: The system can be extended with OpenCV for object detection or quality control[[4]](#fn4)[[5]](#fn5).
* **Additional Robots/Sensors**: Modular design allows more devices to be added via USB or MQTT.
* **Web Dashboard**: Can be enhanced for real-time monitoring, manual override, and analytics.

**9. References**

* [[1]](#fn1) System Design Diagram (provided image)

* [[2]](#fn2) System Flow Diagram (provided image)

* [[4]](#fn4) Industrial Robotic Arm Using Raspberry PI, IJIES, 2024

* [[5]](#fn5) Electron Dust, Raspberry Pi Robot Arm with Simple Computer Vision, 2017

* [[6]](#fn6) RobotShop Community, Raspberry Pi Robot Arm with Simple Computer Vision, 2015

* [[3]](#fn3) danielnitz/DobotPi: Control a Dobot robotic arm with a raspberry Pi

**10. Conclusion**

This design leverages open-source technologies and modular architecture to create a flexible, secure, and extensible M2M system for robotic automation. The integration of MQTT, Node-RED, Docker, and Python ensures robust communication and control, while the use of a webcam provides valuable monitoring and the potential for advanced computer vision features. The system is well-suited for prototyping, educational, or light industrial automation applications.

⁂

1. <https://github.com/danielnitz/DobotPi>

1. <https://www.ijies.net/finial-docs/finial-pdf/170524939.pdf>

1. <https://www.electrondust.com/2017/10/28/raspberry-pi-robot-arm-with-simple-computer-vision/>

1. <https://community.robotshop.com/forum/t/raspberry-pi-robot-arm-with-simple-computer-vision/7079>