Báo cáo Kỹ thuật Chuyên sâu: Mở rộng Kiến trúc Điều khiển Robot 6-DOF Đa phương thức (Welding & Pick-and-Place)

Tổng quan Điều hành

Báo cáo này trình bày một phân tích kỹ thuật toàn diện và lộ trình triển khai nhằm nâng cấp hệ thống robot 6 bậc tự do (6-DOF) hiện hữu từ một trạm hàn chuyên dụng thành một tế bào sản xuất linh hoạt đa chức năng (Universal Mode). Hệ thống hiện tại vận hành trên kiến trúc phân tầng: **Giao diện Người máy (HMI) trên PC (C# WPF)** $\rightarrow$ **Lõi điều khiển thời gian thực (C++ Core sử dụng RL/Ruckig)** $\rightarrow$ **Bộ điều khiển chuyển động nhúng (Teensy 4.1 chạy grblHAL)**. Yêu cầu đặt ra là tích hợp thêm năng lực Thị giác máy tính (Computer Vision) và Trí tuệ nhân tạo (AI) dựa trên Python để thực hiện tác vụ Gắp và Đặt (Pick & Place - P&P).

Sự chuyển dịch từ hàn (Welding) sang gắp đặt (P&P) không chỉ đơn thuần là thay đổi công cụ cuối (End-effector), mà là sự thay đổi căn bản về triết lý điều khiển: từ chuyển động theo quỹ đạo liên tục (Continuous Path - CP) với độ chính xác cao sang chuyển động điểm-điểm (Point-to-Point - PTP) tối ưu hóa thời gian, và từ môi trường có cấu trúc cố định sang môi trường bán cấu trúc dựa trên cảm biến.

Báo cáo này tập trung giải quyết bốn thách thức kỹ thuật cốt lõi:

1. **Kiến trúc Phần mềm:** Lựa chọn giao thức giao tiếp giữa các tiến trình (IPC) để đảm bảo độ trễ thấp (<1ms) cho vòng lặp điều khiển.
2. **Hệ thống Thị giác:** Phân tích sự đánh đổi giữa Camera 2D và 3D trong bối cảnh xử lý lỗi sai biệt thị giác (parallax error).
3. **Hiệu chuẩn Hand-Eye:** Phương pháp toán học để đồng bộ hóa hệ tọa độ thị giác và hệ tọa độ robot.
4. **An toàn và Logic Chuyển đổi:** Thiết kế máy trạng thái hữu hạn (Finite State Machine) để quản lý việc thay đổi công cụ và đảm bảo an toàn vận hành.

---

1. Kiến trúc Phần mềm và Giao thức Giao tiếp (IPC)

Trong hệ thống robot hiện đại, đặc biệt là các hệ thống lai giữa điều khiển thời gian thực (C++/Teensy) và suy luận AI (Python), "nút thắt cổ chai" thường nằm ở lớp trung gian (Middleware). Hệ thống hiện tại đang sử dụng C++ Core làm trung tâm xử lý quỹ đạo (Ruckig). Việc tích hợp Python AI đòi hỏi một cơ chế giao tiếp không làm gián đoạn chu kỳ tính toán (control loop cycle) của C++ Core.

1.1 Phân tích So sánh: ZeroMQ vs. gRPC vs. TCP Raw

Để lựa chọn giao thức tối ưu, chúng ta cần xem xét ba tiêu chí chính: Độ trễ (Latency), Thông lượng (Throughput), và Độ phức tạp tích hợp.

1.1.1 ZeroMQ (ZMQ): Hiệu suất cao và Phi tập trung

ZeroMQ là một thư viện nhắn tin không đồng bộ, cung cấp sự trừu tượng hóa của socket nhưng tự động quản lý việc thiết lập kết nối, xếp hàng tin nhắn (queueing), và đóng gói khung (framing).

- **Độ trễ và Hiệu năng:** Các nghiên cứu thực nghiệm cho thấy ZeroMQ vượt trội hơn gRPC trong các tình huống yêu cầu độ trễ cực thấp. Với giao thức IPC (Inter-Process Communication) hoặc TCP loopback, ZeroMQ có thể đạt độ trễ trung bình từ 30$\mu $sđến100$ \mu$s.[1] Điều này cực kỳ quan trọng đối với C++ Core, nơi mà mỗi mili-giây (ms) đều ảnh hưởng đến độ mượt của chuyển động robot (Jerk).
- **Mô hình Messaging:** ZeroMQ hỗ trợ các mẫu thiết kế (patterns) mạnh mẽ như PUB/SUB (Publish-Subscribe) và REQ/REP (Request-Reply). Mô hình PUB/SUB cho phép C++ Core "phát sóng" trạng thái robot (góc khớp, vận tốc, mô-men xoắn) ở tần số cao (ví dụ: 500Hz) mà không cần quan tâm đến việc bên nhận (Python AI hoặc C# UI) có xử lý kịp hay không. Điều này giúp tách biệt hoàn toàn hiệu năng của lõi điều khiển khỏi sự biến thiên của Python (do Garbage Collection).[4]
- **Khả năng mở rộng:** ZeroMQ hoạt động theo mô hình "Brokerless" (không có máy chủ trung gian), giúp giảm điểm nghẽn và đơn giản hóa kiến trúc mạng cục bộ.[3]

1.1.2 gRPC: Cấu trúc chặt chẽ và Hướng dịch vụ

gRPC, phát triển bởi Google, sử dụng HTTP/2 làm lớp vận chuyển và Protocol Buffers (Protobuf) để tuần tự hóa dữ liệu.

- **Ưu điểm:** gRPC định nghĩa giao diện rất rõ ràng thông qua file .proto. Điều này đảm bảo tính nhất quán của dữ liệu (Type Safety) giữa C#, C++ và Python. Việc gọi một hàm từ xa (Remote Procedure Call) trông giống như gọi hàm cục bộ, giúp code dễ đọc và bảo trì.[6]
- **Nhược điểm:** Cơ chế tuần tự hóa/giải tuần tự hóa (Serialization/Deserialization) của Protobuf và overhead của HTTP/2 tạo ra độ trễ cao hơn đáng kể so với ZeroMQ, thường ở mức vài mili-giây (ms) thay vì micro-giây (μs). Trong vòng lặp điều khiển thời gian thực (Real-time Control Loop), độ trễ này có thể gây ra hiện tượng trượt vị trí hoặc phản ứng chậm với các tín hiệu cảm biến.[1] Ngoài ra, việc phân bổ bộ nhớ động trong quá trình parse Protobuf có thể không tối ưu cho C++ Core hiệu năng cao.

1.1.3 TCP Raw: Kiểm soát tối đa nhưng Phức tạp

Sử dụng socket TCP trần (Raw TCP) mang lại hiệu năng lý thuyết tốt nhất nhưng đòi hỏi lập trình viên phải tự xử lý việc phân mảnh gói tin (fragmentation), tái lập (reassembly), và xử lý lỗi ngắt kết nối. Trong môi trường đa ngôn ngữ (C#/C++/Python), việc duy trì tính ổn định của

TCP Raw tốn kém nhiều nỗ lực phát triển hơn lợi ích nó mang lại so với ZeroMQ.[9]

1.2 Kiến trúc Đề xuất: Mô hình Lai dựa trên ZeroMQ

Dựa trên yêu cầu giữ nguyên hiệu năng của C++ Core (RL/Ruckig) và Teensy 4.1, tôi đề xuất kiến trúc **ZeroMQ-centric** với cấu trúc liên kết mạng (Network Topology) như sau:

Bảng 1: Phân bổ kênh giao tiếp ZeroMQ

Kênh (Channel)	Pattern	Sender (Bên gửi)	Receiver (Bên nhận)	Dữ liệu	Tần số (Hz)	Mục đích
State Stream	PUB/SUB	C++ Core	C# WPF, Python AI	Joint Angles, TCP Pose, Velocities, IO Status	100 - 500 Hz	Cung cấp dữ liệu thời gian thực cho hiển thị và đầu vào suy luận AI.
Command Link	REQ/REP	Python AI / C#	C++ Core	Waypoints, Grip/Release, Mode Change	Bất đồng bộ (Async)	Gửi lệnh di chuyển hoặc thay đổi trạng thái, yêu cầu xác nhận (ACK).
Heartbeat	PUB/SUB	Python AI	C++ Core	Timestamp, Status Code	10 - 20 Hz	Giám sát "sức khỏe" của module AI. Kích hoạt Safety Stop nếu mất tín

						hiệu.
Vision Result	PUSH/PULL	Python AI	C++ Core	Object Coordinates (X, Y, Z, Rz)	Theo sự kiện (Event-driven)	Trả về tọa độ vật thể sau khi xử lý ảnh.

1.2.1 Chiến lược Tuần tự hóa (Serialization Strategy)

Để tối ưu hóa băng thông và CPU:

- **Kênh State Stream:** Sử dụng cấu trúc nhị phân (Binary Struct) của C++ được đóng gói trực tiếp (memcpy). Python sẽ sử dụng module struct để giải mã (unpack) cực nhanh mà không cần parse chuỗi. Điều này giảm tải CPU cho C++ Core gần như bằng 0.[10]
  - *Ví dụ:* struct RobotState { uint64_t timestamp; double joints; double tcp; uint8_t status; }
- **Kênh Command Link:** Sử dụng **JSON**. Vì tần suất gửi lệnh thấp hơn nhiều so với luồng trạng thái, JSON mang lại tính linh hoạt cao, dễ debug, và dễ mở rộng khi thêm các tham số mới cho chế độ AI mà không cần biên dịch lại toàn bộ C++ Core.[10]

1.2.2 Xử lý vấn đề "Slow Subscriber" trong Python

Python, do cơ chế Global Interpreter Lock (GIL) và Garbage Collection (GC), có thể bị "treo" ngẫu nhiên trong vài chục mili-giây. Nếu C++ gửi dữ liệu ở 500Hz, hàng đợi (queue) phía Python sẽ bị đầy, dẫn đến việc xử lý dữ liệu cũ (latency spike).

- **Giải pháp:** Sử dụng tùy chọn ZMQ_CONFLATE trên socket SUB của Python. Tùy chọn này chỉ giữ lại tin nhắn *mới nhất* trong hàng đợi và hủy bỏ các tin nhắn cũ chưa được đọc. Điều này đảm bảo thuật toán AI luôn chạy trên dữ liệu trạng thái hiện tại của robot, loại bỏ độ trễ tích lũy.[12]

1.3 Tích hợp với Teensy 4.1 (grblHAL)

C++ Core đóng vai trò là "Master" gửi lệnh G-code hoặc luồng vị trí thời gian thực xuống Teensy.

- **Giao thức:** USB Serial (Virtual COM Port) ở tốc độ cao (Teensy 4.1 hỗ trợ 480 Mbit/s).
- **Đồng bộ hóa:** C++ Core cần đọc báo cáo trạng thái thời gian thực (? command trong grblHAL) để cập nhật vị trí thực tế của robot vào luồng ZeroMQ. Việc Teensy báo cáo vị trí MPos (Machine Position) là cực kỳ quan trọng để đảm bảo tính chính xác của vòng lặp kín (closed-loop) ảo.[13]

---

2. Hệ thống Thị giác: Lựa chọn Camera và Xử lý Lỗi Parallax

Chế độ "Universal Mode" ngụ ý khả năng xử lý đa dạng các vật thể, có thể là phôi hàn, linh kiện lắp ráp, hoặc sắp xếp ngẫu nhiên. Sự lựa chọn giữa 2D và 3D quyết định giới hạn vật lý của hệ thống.

2.1 Bản chất Vật lý của Camera 2D và Vấn đề Parallax

Camera 2D hoạt động dựa trên mô hình máy ảnh lỗ kim (Pinhole Camera Model), chiếu thế giới 3D (X, Y, Z) lên mặt phẳng cảm biến 2D (u, v).

Phương trình chiếu hình:

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y$$

Trong đó Z là khoảng cách từ tâm quang học đến vật thể.

Lỗi Parallax (Thị sai): Trong một hệ thống 2D tiêu chuẩn, để tính tọa độ thế giới (X, Y) từ điểm ảnh (u, v), chúng ta bắt buộc phải giả định giá trị Z là cố định (ví dụ: $Z = Z_{conveyor}$). Tuy nhiên, nếu vật thể có chiều cao khác nhau (ví dụ: xếp chồng lên nhau hoặc các loại vật thể khác nhau), giá trị Z thực tế sẽ sai lệch so với Z giả định. Điều này dẫn đến sai số vị trí ngang ($\Delta X, \Delta Y$) tỷ lệ thuận với độ lệch chiều cao và góc nhìn.[15]

- *Ví dụ thực tế:* Nếu camera nhìn nghiêng một góc 30 độ so với phương thẳng đứng, một sai số chiều cao 10mm sẽ dẫn đến sai số vị trí gắp khoảng 5.8mm ($10 \cdot \tan(30^\circ)$). Sai số này là quá lớn cho nhiệm vụ hàn hoặc lắp ráp chính xác.[17]

2.2 Công nghệ Camera 3D: Giải pháp cho Môi trường Bán cấu trúc

Camera 3D (sử dụng Stereo Vision, Structured Light, hoặc Time-of-Flight) cung cấp bản đồ độ sâu (Depth Map) hoặc đám mây điểm (Point Cloud) cho từng điểm ảnh.

- **Ưu điểm:** Hệ thống 3D biết chính xác giá trị Z tại mọi điểm. Do đó, nó có thể tính toán chính xác X, Y bất kể chiều cao vật thể hay góc nghiêng bề mặt. Điều này cho phép robot thực hiện các tác vụ phức tạp như "Bin Picking" (gắp vật lộn xộn trong thùng).[19]
- **Nhược điểm:** Chi phí cao hơn, yêu cầu năng lực tính toán lớn hơn để xử lý Point Cloud, và đôi khi gặp khó khăn với bề mặt kim loại phản quang mạnh (thường gặp trong hàn).[21]

2.3 Phân tích Quyết định và Đề xuất Giải pháp

Dựa trên yêu cầu mở rộng từ hệ thống hàn (vốn yêu cầu độ chính xác cao), tôi đề xuất hai phương án tùy thuộc vào ngân sách và độ phức tạp của "Universal Mode":

Phương án A (Khuyên dùng - Hiệu năng cao): Camera 3D Structured Light

- **Công nghệ:** Sử dụng ánh sáng cấu trúc (Structured Light) như các dòng Mech-Mind, Zivid hoặc Photoneo. Công nghệ này cho độ chính xác cao trên bề mặt kim loại (tốt hơn Stereo Vision thụ động) và giải quyết triệt để vấn đề chiều cao biến thiên.
- **Ứng dụng:** Gắp phôi hàn từ thùng chứa ngẫu nhiên, định vị đường hàn trên bề mặt cong, kiểm tra chất lượng mối hàn sau khi hoàn thành.[19]

Phương án B (Chi phí thấp - Giới hạn): Camera 2D + Cảm biến khoảng cách Laser

- **Cấu hình:** Camera 2D độ phân giải cao kết hợp với một cảm biến laser điểm (Laser Displacement Sensor) gắn cạnh camera.
- **Thuật toán bù trừ Parallax:**
  1. Camera 2D phát hiện tâm vật thể (u, v).
  2. Robot di chuyển cảm biến laser đến vị trí tâm đó để đo chiều cao thực tế Z_{real} .
  3. Áp dụng công thức hiệu chỉnh:

$$X_{real} = X_{img} \cdot \left(1 - \frac{\Delta Z}{H_{camera}}\right)$$

- **Lưu ý:** Phương pháp này chậm hơn (cần bước đo laser) nhưng rẻ hơn nhiều so với 3D toàn phần.[23]

---

3. Hiệu chuẩn Hand-Eye: Toán học và Quy trình

Hiệu chuẩn Hand-Eye là quá trình xác định ma trận chuyển đổi cứng (Rigid Transformation Matrix) giữa hệ tọa độ của Camera và hệ tọa độ của Robot (Base hoặc End-Effector). Đây là nền tảng toán học để robot biết "nhìn thấy ở đâu thì gắp ở đó".

3.1 Cấu hình Lắp đặt: Eye-in-Hand vs. Eye-to-Hand

3.1.1 Eye-in-Hand (Camera gắn trên khâu cuối)

- **Mô tả:** Camera được gắn cố định vào mặt bích (flange) trục thứ 6 của robot, di chuyển cùng với mỏ hàn/tay gắp.
- **Ưu điểm:** Linh hoạt, robot có thể đưa camera lại gần để nhìn chi tiết hoặc ra xa để nhìn bao quát. Tránh được vấn đề vật thể bị che khuất bởi chính cánh tay robot. Phù hợp cho việc kiểm tra mối hàn (inspection).[25]
- **Nhược điểm:** Dây cáp phức tạp (phải chạy dọc thân robot). Chu kỳ làm việc (Cycle time)

chậm hơn vì robot phải dừng lại để chụp ảnh ổn định.[25]

- **Ma trận cần tìm:** ${}^{E}T_{C}$ (Từ End-effector sang Camera).

3.1.2 Eye-to-Hand (Camera cố định)

- **Mô tả:** Camera gắn trên khung giá cố định, nhìn xuống vùng làm việc.
- **Ưu điểm:** Chu kỳ nhanh hơn (chụp ảnh và xử lý song song khi robot đang di chuyển). Không ảnh hưởng tải trọng robot.
- **Nhược điểm:** Trường nhìn cố định. Độ phân giải giảm nếu vùng làm việc lớn. Có nguy cơ robot che khuất vật thể khi di chuyển.[26]
- **Ma trận cần tìm:** ${}^{B}T_{C}$ (Từ Base sang Camera).

Đề xuất: Đối với hệ thống lai Hàn/P&P, cấu hình **Eye-to-Hand** thường an toàn hơn để bắt đầu vì tránh việc gắn thiết bị điện tử nhạy cảm (camera) ngay cạnh mỏ hàn tỏa nhiệt và nhiễu điện từ. Tuy nhiên, nếu cần soi đường hàn (Seam Tracking), Eye-in-Hand là bắt buộc. Giả định "Universal Mode" tập trung vào gắp đặt phôi, **Eye-to-Hand** là lựa chọn tối ưu về an toàn và tốc độ.

3.2 Cơ sở Toán học: Giải phương trình $AX = XB$

Quá trình hiệu chuẩn dựa trên việc giải phương trình ma trận kinh điển $AX = XB$ (đối với Eye-in-Hand) hoặc $AX = ZB$ (đối với Eye-to-Hand).

- **A:** Chuyển động của tay máy robot (từ bộ điều khiển). $A_i = T_{base}^{end_i}$.
- **B:** Chuyển động của camera (từ xử lý ảnh bàn cờ). $B_i = T_{camera}^{target_i}$.
- **X:** Ma trận ẩn cần tìm (${}^{E}T_{C}$ hoặc ${}^{B}T_{C}$).

Quy trình thực hiện (Python + OpenCV):

1. **Thu thập dữ liệu:**
   - Sử dụng một bàn cờ chuẩn (ChArUco hoặc Chessboard) cố định (đối với Eye-in-Hand) hoặc gắn trên tay kẹp (đối với Eye-to-Hand).
   - Điều khiển robot di chuyển đến N vị trí khác nhau (khuyên dùng $N \geq 15$). Các vị trí này phải đa dạng về góc xoay (Rotation) và tịnh tiến (Translation). Nếu chỉ tịnh tiến mà không xoay, thuật toán sẽ bị suy biến và không giải được.[28]
   - Tại mỗi vị trí, ghi lại cặp dữ liệu: Pose của Robot (R_{rob}, t_{rob}) từ C++ Core và Pose của Bàn cờ (R_{cam}, t_{cam}) từ camera.
2. **Tính toán:**
   - Sử dụng hàm cv2.calibrateHandEye() trong thư viện OpenCV.

- Các phương pháp giải phổ biến:
  - **Tsai-Lenz:** Hiệu quả và tốc độ cao.
  - **Park-Martin:** Dựa trên chuỗi Lie, thường ổn định hơn với nhiễu.[30]
- *Code snippet (Python):*

```
Python
R_cam2gripper, t_cam2gripper = cv2.calibrateHandEye(
  R_gripper2base, t_gripper2base,
  R_target2cam, t_target2cam,
  method=cv2.CALIB_HAND_EYE_PARK
)
```

3. **Đánh giá sai số:**
   - Sau khi có ma trận X, thực hiện kiểm tra chéo bằng cách đưa robot đến 5 điểm bất kỳ, so sánh tọa độ tính toán từ vision ($P_{robot} = X \cdot P_{camera}$) với vị trí thực tế. Sai số chấp nhận được cho P&P công nghiệp thường là $< 1mm$.[21]

---

4. An toàn Chuyển đổi Chế độ và Logic Điều khiển

Việc chuyển đổi giữa chế độ Hàn (Welding) và Gắp đặt (Universal) chứa đựng rủi ro vật lý lớn. Mỏ hàn đang nóng, điện áp cao, khí bảo vệ; trong khi tay gắp (Gripper) sử dụng khí nén hoặc điện và yêu cầu va chạm vật lý với vật thể.

4.1 Máy Trạng Thái Hữu Hạn (Finite State Machine - FSM)

Hệ thống C++ Core cần cài đặt một FSM nghiêm ngặt để quản lý quá trình này. Không được phép chuyển đổi trực tiếp ("Hot Swap") mà phải qua các bước trung gian an toàn.

Các trạng thái định nghĩa:

1. **STATE_WELDING (Hàn):**
   - Công cụ: Torch.
   - Logic an toàn: Giám sát hồ quang, khí, và nhiệt độ.
   - Điều khiển: Ruckig hoạt động ở chế độ *Path Control* (giữ vận tốc đầu mỏ hàn không đổi).
2. **STATE_CHANGEOVER (Chuyển đổi):**
   - Robot di chuyển về vị trí an toàn (Home).
   - Ngắt điện nguồn hàn (Power Source OFF).
   - Di chuyển đến trạm thay Tool (Tool Changer Stand).
   - Thực hiện trình tự: Thả Torch $\rightarrow$ Xác nhận cảm biến $\rightarrow$ Gắp Gripper $\rightarrow$ Khóa chốt $\rightarrow$ Xác nhận cảm biến.[33]

3. **STATE_UNIVERSAL (Gắp đặt):**
    - Công cụ: Gripper.
    - Logic an toàn: Giới hạn lực (Force Limiting) và tốc độ (Collaborative Speed).
    - Điều khiển: Ruckig chuyển sang chế độ *Time-Optimal* (tối ưu thời gian di chuyển điểm-điểm, chấp nhận Jerk cao hơn).[35]

4.2 Quản lý Bù trừ Chiều dài Dụng cụ (Tool Length Offset - TLO) trong grblHAL

Khi thay đổi từ Mỏ hàn (dài, cong) sang Gripper (ngắn, thẳng), điểm TCP (Tool Center Point) thay đổi hoàn toàn. Nếu không cập nhật TLO, robot sẽ lao đầu kẹp vào bàn làm việc hoặc gắp hụt.

Cơ chế cập nhật trong grblHAL:

- GrblHAL hỗ trợ bảng dao (Tool Table) thông qua các lệnh G-code mở rộng hoặc lưu trong EEPROM.
- **Lệnh G10:** Cài đặt offset. Ví dụ: G10 L1 P1 Z150.5 (Cài đặt dao số 1 có chiều dài Z = 150.5mm).
- **Lệnh M6:** Thay dao. Khi nhận lệnh M6 T1, C++ Core cần gửi chuỗi macro để robot thực hiện động tác thay cơ khí (nếu có ATC) hoặc dừng lại chờ thay tay.
- **Lệnh G43:** Áp dụng bù trừ. G43 H1 sẽ cộng dồn giá trị offset của dao số 1 vào tọa độ Z hiện tại. Điều này cực kỳ quan trọng để code di chuyển (G0/G1) vẫn giữ nguyên giá trị Z mặt bàn (Z=0) bất kể đang gắn công cụ gì.[36]

Chiến lược tích hợp:

Trong C++ Core, khi chuyển Mode, hệ thống sẽ tự động gửi một chuỗi G-code xuống Teensy:

1. G49 (Hủy bù trừ cũ - Safety).
2. M6 Tx (Thực hiện thay dao vật lý).
3. G43 Hx (Áp dụng bù trừ mới).
4. Cập nhật mô hình động học (Kinematics Model) trong Ruckig để tính toán lại giới hạn khớp (Joint Limits) và tránh va chạm (Self-collision) với công cụ mới.

4.3 Cơ chế Heartbeat và Watchdog An toàn

Vì Python AI chạy trên hệ điều hành không thời gian thực (Windows/Linux) và giao tiếp qua mạng, rủi ro mất kết nối hoặc "treo" tiến trình là hiện hữu.

- **Cơ chế:** Python gửi gói tin { "heartbeat": timestamp } qua ZeroMQ PUB mỗi 50ms.
- **Xử lý tại C++ Core:** Nếu không nhận được heartbeat mới trong vòng 200ms (cửa sổ an toàn), C++ Core lập tức kích hoạt chế độ **Soft Stop** (giảm tốc về 0 với gia tốc tối đa cho phép) và khóa các đầu ra (Digital Outputs) điều khiển kẹp/hàn. Điều này ngăn robot tiếp tục di chuyển mù quáng khi "bộ não" AI đã ngừng hoạt động.[39]

5. Lộ trình Triển khai Chi tiết

Giai đoạn 1: Nâng cấp Cơ sở Hạ tầng IPC (Tuần 1-2)

- Tích hợp thư viện cppzmq vào C++ Core và pyzmq vào Python.
- Thiết kế cấu trúc dữ liệu RobotState và ControlCommand.
- Viết Unit Test đo độ trễ "Round-trip" (Python gửi lệnh $\rightarrow$ C++ phản hồi $\rightarrow$ Python nhận). Mục tiêu: < 2ms trên localhost.
- Cài đặt cơ chế Heartbeat Watchdog.

Giai đoạn 2: Tích hợp Hệ thống Thị giác (Tuần 3-4)

- Lắp đặt camera (ưu tiên Eye-to-Hand để thử nghiệm).
- Viết script Python thực hiện quy trình thu thập dữ liệu hiệu chuẩn tự động (Gửi lệnh di chuyển qua ZeroMQ $\rightarrow$ Chụp ảnh $\rightarrow$ Lưu dữ liệu).
- Chạy thuật toán Hand-Eye Calibration và kiểm chứng sai số thực tế bằng cách chỉ định robot chạm vào một điểm đánh dấu.

Giai đoạn 3: Mở rộng grblHAL và Safety Logic (Tuần 5-6)

- Đo đạc chính xác kích thước hình học (TCP offsets) của Mỏ hàn và Gripper.
- Cập nhật bảng dao trong grblHAL hoặc lưu trữ offsets trong file cấu hình JSON của C++ Core để nạp động (G10 L1).
- Lập trình Macro thay dao (nếu dùng thay dao tự động) hoặc quy trình dừng chờ thay tay (nếu thủ công).
- Test quy trình chuyển đổi trạng thái FSM: Welding $\rightarrow$ Transition $\rightarrow$ PnP $\rightarrow$ Error.

Giai đoạn 4: Hoàn thiện AI và Vận hành Thử nghiệm (Tuần 7-8)

- Huấn luyện mô hình nhận diện vật thể (ví dụ: YOLOv8 hoặc 3D Point Cloud Segmentation).
- Tích hợp logic "Look-then-Move": Camera chụp $\rightarrow$ AI tính toán tọa độ $\rightarrow$ Chuyển đổi sang Robot Base Frame $\rightarrow$ Gửi lệnh tới C++ Core.
- Tinh chỉnh tham số Ruckig cho chế độ PnP để đạt tốc độ tối đa mà không gây rung động.

6. Kết luận

Việc mở rộng hệ thống robot 6-DOF từ hàn sang đa năng là một bài toán phức tạp về tích hợp hệ thống hơn là phát minh thuật toán mới. Chìa khóa thành công nằm ở sự **phân tách rõ ràng** (Decoupling) giữa lớp điều khiển cứng (C++/Teensy) và lớp trí tuệ mềm (Python/AI) thông qua ZeroMQ, cùng với sự **chính xác tuyệt đối** trong mô hình toán học (Hand-Eye Calibration, TCP

Offsets). Bằng cách tuân thủ kiến trúc đề xuất này, hệ thống sẽ đạt được sự linh hoạt của AI hiện đại mà không hy sinh độ tin cậy và an toàn vốn có của một robot công nghiệp.

Works cited

1. Comparative Analysis OF GRPC VS. ZeroMQ for Fast Communication - Jetir.Org, accessed February 1, 2026, https://www.jetir.org/papers/JETIR2002540.pdf
2. ZeroMQ: High-Performance Concurrency Framework - Hacker News, accessed February 1, 2026, https://news.ycombinator.com/item?id=40964852
3. Comparison of ZeroMQ and Redis for a robot control platform - GitHub Gist, accessed February 1, 2026, https://gist.github.com/hmartiro/85b89858d2c12ae1a0f9
4. Chapter 5 - Advanced Pub-Sub Patterns - ZeroMQ Guide, accessed February 1, 2026, https://zguide.zeromq.org/docs/chapter5/
5. Pub-Sub - NetMQ - Read the Docs, accessed February 1, 2026, https://netmq.readthedocs.io/en/latest/pub-sub/
6. Comparative Analysis OF GRPC VS. ZeroMQ for Fast Communication - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/389078536_Comparative_Analysis_OF_GRPC_VS_ZeroMQ_for_Fast_Communication
7. What ZeroMQ gRPC Actually Does and When to Use It - hoop.dev, accessed February 1, 2026, https://hoop.dev/blog/what-zeromq-grpc-actually-does-and-when-to-use-it/
8. Comparing gRPC performance across different technologies - Nexthink, accessed February 1, 2026, https://nexthink.com/blog/comparing-grpc-performance
9. grpc and zeromq comparsion [closed] - Stack Overflow, accessed February 1, 2026, https://stackoverflow.com/questions/39350681/grpc-and-zeromq-comparsion
10. Designing Scalable JSON Schemas For Computer Vision Pipelines - Medium, accessed February 1, 2026, https://medium.com/@noel.benji/designing-scalable-json-schemas-for-computer-vision-pipelines-dcddf4e7a9f4
11. A Media Type for Describing JSON Documents - JSON Schema, accessed February 1, 2026, https://json-schema.org/draft/2020-12/json-schema-core
12. ZeroMQ PUB/SUB filtering and performance - python - Stack Overflow, accessed February 1, 2026, https://stackoverflow.com/questions/9939238/zeromq-pub-sub-filtering-and-performance
13. grblHAL core code and master Wiki - GitHub, accessed February 1, 2026, https://github.com/grblHAL/core
14. How to register and retrieve tool height data in tool table for the machine with ATC (using macro) and touch off probe at fixed location? · Issue #332 · grblHAL/core - GitHub, accessed February 1, 2026, https://github.com/grblHAL/core/issues/332

15. 1.2. The Pinhole Camera Matrix - Homepages of UvA/FNWI staff, accessed February 1, 2026, https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20162017/LectureNotes/CV/PinholeCamera/PinholeCamera.html
16. Pinhole Camera Model | HediVision, accessed February 1, 2026, https://hedivision.github.io/Pinhole.html
17. How to Select the Right Camera for Pick and Place Robots - Edge AI and Vision Alliance, accessed February 1, 2026, https://www.edge-ai-vision.com/2025/06/how-to-select-the-right-camera-for-pick-and-place-robots/
18. What is the role of cameras in pick and place robots? - e-con Systems, accessed February 1, 2026, https://www.e-consystems.com/blog/camera/applications/what-is-the-role-of-cameras-in-pick-and-place-robots/
19. 2D vs 3D Vision System: Finding a Perfect Fit For Your Application - Photoneo, accessed February 1, 2026, https://www.photoneo.com/2d-vs-3d-vision-system/
20. 2D vs 3D vision: Which is Better For You? - Mech-Mind, accessed February 1, 2026, https://www.mech-mind.com/blog/2d-vs-3d-vision-which-is-better-for-you.html
21. Achieve Optimal Hand-Eye Calibration for Enhanced Robotics Performance - Zivid Blog, accessed February 1, 2026, https://blog.zivid.com/achieving-optimal-hand-eye-calibration-for-enhanced-robotics-performance
22. Bin Picking for Automated Machine Tending | Basler AG, accessed February 1, 2026, https://www.baslerweb.com/en-us/use-cases/bin-picking-machine-tending/wp/
23. 2D Vision vs 3D Vision - Inbolt, accessed February 1, 2026, https://www.inbolt.com/resources/blog/2d-vision-vs-3d-vision/
24. Getting Correct Position of Image based on 2D Parallax Camera - Stack Overflow, accessed February 1, 2026, https://stackoverflow.com/questions/39714288/getting-correct-position-of-image-based-on-2d-parallax-camera
25. [Q] Eye-in-hand vs Eye-to-hand setting : r/robotics - Reddit, accessed February 1, 2026, https://www.reddit.com/r/robotics/comments/eq09rm/q_eyeinhand_vs_eyetohand_setting/
26. Hand-Eye Calibration in Eye to Hand Setup 2.0 (Standard Interface Communication), accessed February 1, 2026, https://www.youtube.com/watch?v=RtaXhcEJyCU
27. Hand-Eye Calibration in Eye to Hand Setup (Standard Interface Communication) - YouTube, accessed February 1, 2026, https://www.youtube.com/watch?v=GnZNOIsaIJ8
28. Hand-Eye Calibration Process — ZIVID KNOWLEDGE BASE documentation, accessed February 1, 2026, https://support.zivid.com/academy/applications/hand-eye/hand-eye-calibration-p

rocess.html
29. zivid-python-samples/source/applications/advanced/hand_eye_calibration/ur_hand_eye_calibration/universal_robots_perform_hand_eye_calibration.py at master - GitHub, accessed February 1, 2026, https://github.com/zivid/python-samples/blob/master/source/applications/advanced/hand_eye_calibration/ur_hand_eye_calibration/universal_robots_perform_hand_eye_calibration.py
30. Camera Calibration and 3D Reconstruction - OpenCV Documentation, accessed February 1, 2026, https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html
31. calibrateHandEye() Python openCV - GeeksforGeeks, accessed February 1, 2026, https://www.geeksforgeeks.org/python/calibratehandeye-python-opencv/
32. A Perspective Distortion Correction Method for Planar Imaging Based on Homography Mapping - MDPI, accessed February 1, 2026, https://www.mdpi.com/1424-8220/25/6/1891
33. Robotic tool changer with integrated safety - Robot System Products, accessed February 1, 2026, https://robotsystemproducts.com/robotic-tool-changer-with-integrated-safety/
34. Automatic / Robotic Tool Changers - ATI Industrial Automation, accessed February 1, 2026, https://www.ati-ia.com/products/toolchanger/robot_tool_changer.aspx
35. One Object at a Time: Accurate and Robust Structure From Motion for Robots - arXiv, accessed February 1, 2026, https://arxiv.org/pdf/2208.00487
36. G49 G-Code [Tool Length Compensation Cancel] - CNC Cookbook, accessed February 1, 2026, https://www.cnccookbook.com/g49-g-code-tool-length-compensation-cancel/
37. G43, G43.1, G49 (Tool Length Offset) - Software, accessed February 1, 2026, https://docs.edingcnc.com/g43-g431-g49-tool-length-offset
38. "Automatic" Tool Length Offset - Unsupported - Carbide 3D Community Site, accessed February 1, 2026, https://community.carbide3d.com/t/automatic-tool-length-offset/8480
39. Heartbeat | voraus Robot Arm 1.4.1 documentation, accessed February 1, 2026, https://docs.vorausrobotik.com/voraus-robot-arm-py/1.4.1/user_guide/heartbeat.html
40. Exploring ZeroMQ's Request Reply Pattern | ICS - Integrated Computer Solutions, accessed February 1, 2026, https://www.ics.com/blog/exploring-zeromqs-request-reply-pattern