BÁO CÁO NGHIÊN CỨU KỸ THUẬT CHUYÊN SÂU: PHÁT TRIỂN HỆ THỐNG ĐIỀU KHIỂN ROBOT HÀN 6 TRỤC TRÊN NỀN TẢNG PC/WPF

1. TỔNG QUAN ĐIỀU HÀNH VÀ PHẠM VI DỰ ÁN

1.1. Bối cảnh và Mục tiêu Kỹ thuật

Báo cáo này được biên soạn nhằm cung cấp một lộ trình kỹ thuật chi tiết và toàn diện cho việc phát triển phần mềm điều khiển robot hàn công nghiệp 6 trục, vận hành trên nền tảng máy tính cá nhân (PC) sử dụng hệ điều hành Windows 10/11. Dự án đặt ra các yêu cầu khắt khe về mặt công nghệ bao gồm việc sử dụng ngôn ngữ C# với nền tảng Windows Presentation Foundation (WPF), tuân thủ mô hình kiến trúc Model-View-ViewModel (MVVM).

Mục tiêu tối thượng của dự án là tái hiện trải nghiệm người dùng (UX) và độ tin cậy vận hành của thiết bị dạy lệnh (teach pendant) tiêu chuẩn công nghiệp **KUKA KRC4 smartPAD**. Đây là một thách thức lớn về mặt kỹ thuật, bởi việc chuyển đổi từ một hệ thống nhúng chuyên dụng (như KUKA KRC4 chạy trên nền tảng thời gian thực VxWorks kết hợp Windows nhúng) sang một môi trường PC thuần túy đòi hỏi sự xử lý tinh tế về độ trễ, đồng bộ luồng dữ liệu và thiết kế giao diện người dùng (UI).[1]

1.2. Phân tích Yêu cầu Cốt lõi

Dựa trên yêu cầu của chủ đầu tư, hệ thống cần đáp ứng bốn trụ cột chính:

1. **Giao diện "Công nghiệp" (Industrial UI):** Không chỉ là thẩm mỹ, mà là sự tối ưu hóa cho màn hình cảm ứng Full HD trong môi trường nhà máy khắc nghiệt, đảm bảo tính dễ đọc và thao tác chính xác.
2. **Inline Forms (Biểu mẫu nội dòng):** Cơ chế lập trình đặc trưng của KUKA, cho phép người dùng không chuyên về code có thể lập trình robot thông qua các biểu mẫu điền tham số trực quan ngay trong trình soạn thảo mã.[3]
3. **Mô phỏng 3D (Helix Toolkit):** Tích hợp môi trường 3D thời gian thực để hiển thị động học robot (Kinematics), kiểm tra va chạm và trực quan hóa đường hàn.
4. **Jogging An toàn:** Cơ chế điều khiển tay robot (Jogging) trên màn hình cảm ứng điện dung thiếu vắng các nút vật lý và công tắc an toàn (Deadman switch) truyền thống.

Báo cáo này sẽ đi sâu phân tích từng khía cạnh, từ kiến trúc phần mềm tầng thấp đến các quyết định thiết kế tầng cao, nhằm đảm bảo sản phẩm cuối cùng không chỉ hoạt động được

mà còn đạt tiêu chuẩn thương mại hóa.

---

2. KIẾN TRÚC HỆ THỐNG PHẦN MỀM (SYSTEM ARCHITECTURE)

2.1. Lựa chọn Công nghệ: Tại sao là WPF và MVVM?

Trong lĩnh vực tự động hóa công nghiệp trên nền tảng Windows, **WPF (Windows Presentation Foundation)** vẫn giữ vị thế độc tôn nhờ khả năng tùy biến giao diện mạnh mẽ qua XAML, hệ thống Data Binding ưu việt và khả năng tận dụng tăng tốc phần cứng (DirectX).

Mô hình MVVM (Model-View-ViewModel) là bắt buộc đối với dự án quy mô này vì những lý do sau [4]:

- **Tách biệt mối quan tâm (Separation of Concerns):** Logic điều khiển robot (Model) hoạt động độc lập hoàn toàn với giao diện hiển thị (View). Điều này cho phép thay đổi giao diện (ví dụ: đổi theme màu, bố cục lại nút bấm) mà không rủi ro làm hỏng logic vận hành cốt lõi.
- **Khả năng kiểm thử (Testability):** Có thể viết Unit Test cho ViewModel (ví dụ: kiểm tra logic giới hạn tốc độ khi Jogging) mà không cần khởi chạy giao diện người dùng.
- **Quản lý trạng thái phức tạp:** Robot công nghiệp có hàng trăm trạng thái (vị trí 6 trục, trạng thái I/O, lỗi, nhiệt độ motor). MVVM giúp đồng bộ hóa các trạng thái này lên giao diện một cách tự động thông qua INotifyPropertyChanged.

2.2. Thách thức Thời gian thực trên Windows (The Real-Time Challenge)

Windows 10/11 không phải là Hệ điều hành thời gian thực (RTOS). Một tác vụ cập nhật Windows Update hoặc trình diệt virus quét nền có thể gây ra độ trễ (latency) lên tới hàng trăm mili-giây. Trong điều khiển robot, chu kỳ nội suy thường yêu cầu độ ổn định 1ms đến 4ms.

Giải pháp Kiến trúc Đề xuất:

Hệ thống phần mềm trên PC sẽ đóng vai trò là **HMI (Human Machine Interface) và Supervisory Control**, không phải là bộ điều khiển chuyển động trực tiếp (Motion Controller). Kiến trúc hệ thống sẽ được phân tầng như sau:

Tầng (Layer)	Thành phần	Vai trò	Công nghệ/Giao thức
HMI Layer	Ứng dụng WPF (PC)	Giao diện, soạn	C#.NET 6/8, Helix

		thảo chương trình, 3D, gửi lệnh cấp cao.	Toolkit
Communication	Middleware	Truyền tải dữ liệu lệnh và trạng thái.	TCP/IP, Modbus TCP, hoặc ADS (Beckhoff)
Control Layer	Motion Controller / PLC	Nội suy quỹ đạo, vòng lặp PID, xử lý an toàn thời gian thực.	SoftPLC (TwinCAT, CODESYS) hoặc Chip vi điều khiển chuyên dụng (STM32/FPGA)

Lưu ý quan trọng: Báo cáo này tập trung vào việc xây dựng **HMI Layer** trên PC. Giả định rằng đã có một bộ điều khiển chuyển động (Motion Controller) bên dưới tiếp nhận lệnh.

2.3. Quản lý Luồng (Threading Model) trong WPF

Một sai lầm phổ biến khi làm ứng dụng công nghiệp bằng WPF là thực hiện xử lý dữ liệu ngay trên luồng giao diện (UI Thread). Với robot gửi dữ liệu vị trí tần suất 100Hz (10ms/gói tin), việc cập nhật UI trực tiếp sẽ làm "đơ" giao diện.

Chiến lược xử lý:

1. **Luồng giao tiếp (Communication Thread):** Chạy ngầm (Background Task), liên tục poll dữ liệu từ Controller và deserialize các gói tin.
2. **Mô hình Dữ liệu (Data Model):** Dữ liệu thô được cập nhật vào một biến thread-safe (ví dụ: Interlocked.Exchange cho các giá trị đơn hoặc ConcurrentQueue cho log).
3. **Cơ chế Throttling (Điều tiết):** UI chỉ có thể hiển thị tối đa 60 khung hình/giây. ViewModel cần sử dụng bộ định thời (Timer) hoặc Reactive Extensions (Rx.NET) để lấy mẫu dữ liệu từ Model và cập nhật lên View với tần suất phù hợp mắt người (ví dụ: 30Hz), tránh làm quá tải bộ thu gom rác (Garbage Collector) và UI Dispatcher.

---

3. NGÔN NGỮ THIẾT KẾ UX/UI CÔNG NGHIỆP (INDUSTRIAL DESIGN LANGUAGE)

Để đạt được phong cách "KUKA KRC4 smartPAD", chúng ta cần giải mã ngôn ngữ thiết kế của thiết bị này. Giao diện công nghiệp không ưu tiên sự bóng bẩy (fancy) mà ưu tiên **Tính Nhận Diện (Readability)**, **Độ Tương Phản (Contrast)** và **An Toàn (Safety)**.[1]

3.1. Phân tích Cấu trúc Màn hình (Layout Anatomy)

Màn hình điều khiển robot tiêu chuẩn thường được chia thành các vùng chức năng cố định (Fixed Zones) để tạo phản xạ cơ bắp cho người vận hành. Trên màn hình Full HD (1920x1080), bố cục đề xuất như sau:

Vùng	Vị trí	Kích thước	Chức năng	Phong cách KUKA
Status Bar	Trên cùng	Cao 60-80px	Hiển thị chế độ (T1/AUT), Trạng thái Motor (O/I), Tốc độ override, Đồng hồ.	Nền xám đậm (#333333), Icon trạng thái phẳng, Text trắng.
Left Sidebar	Trái	Rộng 60-80px	Menu Hamburger, truy cập nhanh các màn hình chức năng (Config, Diagnosis).	Nền xám vừa (#555555), Icon lớn dạng Line Art.
Main Content	Giữa	Linh hoạt	Khu vực làm việc chính: Trình soạn thảo Code hoặc Màn hình 3D.	Nền sáng hơn (#EFEFEF hoặc #2D2D30 tùy theme Sáng/Tối).
Right Softkeys	Phải	Rộng 100-120px	Các phím chức năng thay đổi theo ngữ cảnh (Context-sensitive).	Nền xám, viền nổi 3D nhẹ để giả lập phím vật lý.
Message Window	Dưới (Overlay)	Cao 150px	Cửa sổ thông báo lỗi, cảnh	Nền trong suốt hoặc Đỏ/Vàng

			báo (Pop-up khi cần).	tùy mức độ nghiêm trọng.
SmartKeys Bar	Dưới cùng	Cao 100-120px	Các nút lệnh thao tác trực tiếp: TouchUp, Change Tool, Open Gripper.	Các nút kích thước lớn (min 80x80px) để dễ bấm.

3.2. Bảng Màu và Tương Phản (Color Palette & Contrast)

Hệ thống màu sắc cần tuân thủ quy chuẩn an toàn công nghiệp và nhận diện thương hiệu.[6]

- **Màu Nền Chính (Dark Theme):** #2D2D30 hoặc #333333. Giảm mỏi mắt cho công nhân làm việc ca đêm hoặc trong xưởng thiếu sáng.
- **Màu Thương Hiệu (KUKA Orange):** #FF5E00. Sử dụng làm điểm nhấn (Accent) cho tiêu đề, nút đang được chọn (Active state), hoặc viền focus.
- **Màu Trạng Thái An Toàn:**
  - **Xanh lá (#28A745):** Hệ thống sẵn sàng, Motor On, Chương trình đang chạy.
  - **Đỏ (#DC3545):** Lỗi (Fault), Dừng khẩn cấp (E-Stop), Motor Off.
  - **Vàng (#FFC107):** Cảnh báo, Chế độ T1 (Manual), Override tốc độ thấp.
  - **Xám (#6C757D):** Vô hiệu hóa (Disabled), Chưa kết nối.

3.3. Typography và Kích thước Khả dụng

Trên màn hình cảm ứng công nghiệp, người dùng thường đứng cách xa màn hình 0.5m - 1m và có thể đeo găng tay bảo hộ.

- **Font chữ:** Sử dụng Font không chân (Sans-serif) như **Segoe UI** (mặc định Windows) hoặc **Roboto/DIN** để tạo cảm giác kỹ thuật.
- **Kích thước:**
  - Body text: Tối thiểu 14pt (18px).
  - Header: Tối thiểu 20pt.
  - Số liệu quan trọng (Tọa độ): Tối thiểu 24pt, Font Monospace (Consolas) để các con số thẳng hàng dễ so sánh.
- **Mục tiêu chạm (Touch Target):** Kích thước tối thiểu cho một nút bấm phải là **48x48px** (khoảng 9-10mm vật lý). Các nút quan trọng như "JOG" hay "START" cần kích thước tối thiểu **80x80px**.[8]

---

4. CÔNG NGHỆ INLINE FORMS: TRÁI TIM CỦA LẬP TRÌNH KUKA

Yêu cầu về "Inline Forms" là đặc điểm kỹ thuật phức tạp nhất nhưng cũng đắt giá nhất của hệ

thống này. Nó biến việc viết code phức tạp thành việc điền biểu mẫu đơn giản.[1]

4.1. Cơ chế Hoạt động (Underlying Mechanism)

Trong KUKA KRL (KUKA Robot Language), một lệnh di chuyển PTP (Point-to-Point) thực tế có thể trông như sau trong file .src:

```
Code snippet
```

```
;FOLD PTP P1 Vel=100 % PDAT1 Tool Base;%{PE}%R 8.2.20,...
PTP P1 Vel=100% PDAT1
;ENDFOLD
```

Người dùng không nhìn thấy đoạn code thô này. Họ nhìn thấy một "Block" đồ họa có các ô nhập liệu (Dropdown, Textbox).

- **Logic:** Hệ thống cần một bộ **Parser (Bộ phân tích cú pháp)** hai chiều.
  - *Read:* Đọc file text -> Nhận diện cặp thẻ ;FOLD -> Chuyển đổi thành Đối tượng C# (PtpCommandViewModel).
  - *Write:* Người dùng sửa trên UI -> Serialize đối tượng C# thành chuỗi text KRL đúng cú pháp -> Ghi xuống file.

4.2. Triển khai Kỹ thuật trên WPF (Implementation Strategy)

Để hiển thị danh sách các lệnh này, không thể dùng RichTextBox tiêu chuẩn vì khó nhúng các control phức tạp (ComboBox, Button) vào giữa dòng text một cách mượt mà.

Giải pháp: Sử dụng ItemsControl tùy biến.

4.2.1. Thiết kế Model

Tạo một kiến trúc lớp đa hình (Polymorphic Class Structure) cho các lệnh:

```
C#
```

```
public abstract class ProgramCommandBase : ViewModelBase { }

public class PtpMotionCommand : ProgramCommandBase
{
```

```
    public string PointName { get; set; } // Ví dụ: "P1"
    public double Velocity { get; set; }  // Ví dụ: 100
    public string ApproxLevel { get; set; } // Ví dụ: "C_DIS"
    public int ToolIndex { get; set; }
    public int BaseIndex { get; set; }
}

public class LogicCommand : ProgramCommandBase
{
    public string Variable { get; set; }
    public string Operator { get; set; }
    public string Value { get; set; }
}
```

4.2.2. Thiết kế View (XAML) với TemplateSelector

Sử dụng DataTemplateSelector để chọn giao diện hiển thị dựa trên loại lệnh.[10]

- **PtpTemplate:** Hiển thị một Border bo góc, màu nền xám nhạt. Bên trong là StackPanel ngang chứa:
  - TextBlock "PTP" (Bold, màu Cam).
  - HyperlinkButton bind vào PointName. Khi click sẽ mở popup dạy điểm.
  - ComboBox (đã style lại cho gọn) bind vào Velocity.
  - TextBlock "CONT" nếu có xấp xỉ (C_DIS).
- **LogicTemplate:** Hiển thị cấu trúc IF...THEN với các Dropdown chọn biến số.

4.2.3. Trải nghiệm Người dùng (Interaction Flow)

Khi người dùng bấm nút "Motion" trên thanh SmartKeys:

1. Hệ thống chèn một PtpMotionCommand mới vào ObservableCollection của chương trình hiện tại.
2. Giao diện tự động render một Inline Form mới.
3. Người dùng bấm vào tên điểm "P1" -> Hiện cửa sổ Pop-up "Touch Up?".
4. Người dùng xác nhận -> Hệ thống lấy tọa độ hiện tại của robot từ Model -> Gán vào dữ liệu của điểm P1.
5. Dữ liệu điểm (X, Y, Z, A, B, C) được lưu vào file .dat (theo cấu trúc KUKA) chứ không phải file .src.

---

5. KỸ THUẬT ĐỒ HỌA 3D VỚI HELIX TOOLKIT

Yêu cầu hiển thị 3D và mô phỏng Jogging đòi hỏi một engine đồ họa mạnh mẽ. **Helix Toolkit** là thư viện mã nguồn mở tốt nhất cho WPF 3D.[11]

5.1. Lựa chọn Phiên bản: SharpDX vs. Wpf

Cần phân biệt rõ hai phiên bản của Helix Toolkit [12]:

- HelixToolkit.Wpf: Sử dụng System.Windows.Media.Media3D (DirectX 9). Dễ dùng nhưng hiệu năng thấp, khó render mô hình phức tạp mượt mà.
- HelixToolkit.Wpf.SharpDX: Sử dụng SharpDX (DirectX 11). Hiệu năng cao hơn gấp nhiều lần, hỗ trợ Shader tùy biến. **Đây là lựa chọn bắt buộc** cho dự án này để đảm bảo FPS ổn định trên màn hình Full HD.

5.2. Xây dựng Cây Động học (Kinematic Chain)

Để mô phỏng robot 6 trục, không thể chỉ load một file 3D nguyên khối. Cần load 6 file STL/OBJ riêng biệt tương ứng với 6 khâu (Link) của robot (Base, Link 1, Link 2... Link 6).

Cấu trúc Scene Graph:

Trong Helix Viewport3DX, ta xây dựng một cây phân cấp (Hierarchy Tree):

- SceneNodeGroup (Root)
  - MeshNode (Base - Cố định)
    - SceneNode (Joint 1 - Xoay quanh trục Z)
      - MeshNode (Link 1)
        - SceneNode (Joint 2 - Xoay quanh trục Y)
          - MeshNode (Link 2)
            - ... và tiếp tục đến Flange.

Cập nhật Chuyển động:

ViewModel sẽ bind các thuộc tính Joint1Angle, Joint2Angle... tới thuộc tính Transform (MatrixTransform3D) của các SceneNode. Khi nhận dữ liệu từ Controller, các góc khớp thay đổi -> Ma trận biến đổi được cập nhật -> Robot di chuyển trên màn hình mượt mà.

5.3. Hiển thị Đường dẫn và Vùng làm việc

Để tăng tính "Công nghiệp", hệ thống 3D cần hỗ trợ:

- **TCP Trace:** Vẽ một đường line (PolyLine3D) đi theo chuyển động của đầu mỏ hàn để người dùng hình dung quỹ đạo.
- **Coordinate Systems:** Hiển thị các mũi tên RGB (Đỏ/Xanh/Lam tương ứng X/Y/Z) tại gốc Base và tại đầu Tool (TCP) để hỗ trợ người dùng khi Jogging theo hệ tọa độ.[14]
- **Bóng mờ (Ghost Robot):** Khi người dùng mô phỏng chương trình, hiển thị một robot bán trong suốt chạy trước robot thật để kiểm tra va chạm.

---

6. CƠ CHẾ JOGGING AN TOÀN (SAFETY JOGGING)

Jogging (chạy tay) robot trên màn hình cảm ứng PC là tính năng rủi ro nhất. Thiết bị chuyên dụng như SmartPAD có nút E-Stop cứng và công tắc Deadman 3 vị trí. PC không có những thứ này. Do đó, phần mềm phải thiết kế các lớp bảo vệ thay thế (Mitigation Layers).[15]

6.1. Nguyên tắc "Deadman Ảo" (Software Deadman)

Không bao giờ sử dụng cơ chế "Toggle" (Bấm 1 lần chạy, bấm lần nữa dừng) cho việc Jogging. Phải sử dụng cơ chế **"Hold-to-Run"** (Giữ để chạy).

Triển khai WPF:

Sử dụng sự kiện PreviewMouseLeftButtonDown (hoặc TouchDown) và PreviewMouseLeftButtonUp (hoặc TouchUp/LostMouseCapture).

- **Khi nhấn giữ:** Gửi lệnh JOG_START(Axis, Direction) liên tục hoặc gửi một lần kèm theo nhịp tim (heartbeat).
- **Khi thả tay (hoặc trượt ngón tay ra ngoài nút):** Gửi ngay lập tức lệnh JOG_STOP.

6.2. Bảo vệ Đa lớp (Defense in Depth)

Lớp bảo vệ	Cơ chế	Mô tả kỹ thuật
Lớp 1: Giao diện	Two-Handed Operation	Yêu cầu người dùng nhấn giữ một nút "Enable" bằng tay trái (góc dưới trái màn hình) thì các nút Jog bên phải mới kích hoạt (Enabled). Điều này giả lập công tắc Deadman và buộc người dùng tập trung.
Lớp 2: Logic App	Watchdog Timer (Heartbeat)	Ứng dụng gửi gói tin "Heartbeat" (ví dụ: 0xAA) mỗi 50ms xuống Controller. Nếu Controller không nhận được gói tin này trong 150ms (do Windows bị treo, app crash), Controller tự động ngắt motor.
Lớp 3: Vận tốc	T1 Mode Limit	Khi ở chế độ T1 (Teaching), phần mềm giới hạn thanh trượt tốc độ (Override) tối

		đa ở mức 10% hoặc 250mm/s. Giao diện phải hiển thị cảnh báo nếu người dùng cố kéo cao hơn.

6.3. Xử lý Cảm ứng Đa điểm và "Ghost Touch"

Màn hình cảm ứng công nghiệp đôi khi bị nhiễu (do tia lửa hàn, bụi bẩn) gây ra các điểm chạm ảo (Ghost Touch).

- **Bộ lọc (Filter):** Cài đặt ngưỡng nhận diện điểm chạm. Điểm chạm phải tồn tại ít nhất 50ms mới được coi là hợp lệ.
- **Logic loại trừ:** Nếu phát hiện 2 lệnh Jog đối nghịch (ví dụ: J1+ và J1-) được nhấn cùng lúc (do lỗi cảm ứng hoặc người dùng vô tình tỳ tay), hệ thống phải ưu tiên **DỪNG** ngay lập tức.

---

7. TÍCH HỢP CÔNG NGHỆ HÀN (WELDING TECH PACKAGES)

Robot hàn khác robot gắp thả ở chỗ nó quản lý quy trình công nghệ hàn (Arc Welding Process).

7.1. Các tham số chuyên biệt (Weld Parameters)

Giao diện cần có các Inline Forms chuyên dụng cho lệnh hàn:

- ARC_START (WeldID, SeamType, IgnitionParam)
- ARC_END (CraterFillTime, BurnBackTime)
- WEAVE (Pattern, Amplitude, Frequency) - Lệnh dao động mỏ hàn (Weaving).

7.2. Giám sát Thời gian thực (Live Scope)

Trong quá trình hàn, người dùng cần quan sát sự ổn định của hồ quang. Sử dụng thư viện biểu đồ hiệu năng cao (như **OxyPlot** hoặc **LiveCharts Geared**) để vẽ đồ thị cuộn (Rolling Chart) theo thời gian thực cho:

- Điện áp hàn (Voltage - V)
- Dòng điện hàn (Current - A)
- Tốc độ cấp dây (Wire Feed Speed).

Mô hình MVVM cần xử lý việc đẩy dữ liệu này từ Model lên View một cách mượt mà, tránh vẽ lại toàn bộ biểu đồ gây giật lag. Sử dụng cơ chế bộ đệm vòng (Circular Buffer) để lưu trữ 1000

điểm dữ liệu gần nhất cho việc hiển thị.

---

8. KẾT LUẬN VÀ KHUYẾN NGHỊ TRIỂN KHAI

Việc phát triển phần mềm điều khiển robot hàn 6 trục trên PC với phong cách KUKA KRC4 là một dự án đầy tham vọng nhưng hoàn toàn khả thi với công nghệ C# WPF hiện đại. Chìa khóa thành công nằm ở:

1. **Kiến trúc:** Tuân thủ nghiêm ngặt MVVM để quản lý sự phức tạp.
2. **Đồ họa:** Sử dụng Helix Toolkit SharpDX để đảm bảo hiệu năng 3D.
3. **UX/UI:** Sao chép không chỉ màu sắc mà cả triết lý vận hành (Inline Forms, SmartKeys) của KUKA.
4. **An toàn:** Xây dựng các cơ chế an toàn phần mềm (Watchdog, Two-handed operation) để bù đắp cho sự thiếu hụt phần cứng chuyên dụng trên PC.

Báo cáo này cung cấp nền tảng lý thuyết và hướng dẫn thực hành chi tiết để đội ngũ phát triển bắt tay vào xây dựng bản mẫu (Prototype) đầu tiên.

Lưu ý: Báo cáo này được biên soạn dựa trên các nguyên tắc thiết kế phần mềm công nghiệp tốt nhất và các tài liệu nghiên cứu về hệ thống KUKA, WPF và Helix Toolkit. Việc triển khai thực tế cần tuân thủ các quy định an toàn lao động địa phương và tiêu chuẩn ISO 10218 về an toàn robot công nghiệp.

Works cited

1. KUKA.UserTech – GUI for easy robot configuration | KUKA Global, accessed February 1, 2026, https://www.kuka.com/en-us/company/press/news/2023/04/kuka-usertech-human-robot-interface
2. KUKA smartPAD solutions: Ergonomic teach pendants for robots, accessed February 1, 2026, https://www.kuka.com/en-us/products/robotics-system/robot-controllers/smartpad-robot-teach-pendant
3. KUKA.UserTech – GUI for easy robot configuration, accessed February 1, 2026, https://www.kuka.com/en-de/company/press/news/2023/04/kuka-usertech-human-robot-interface
4. Patterns - WPF Apps With The Model-View-ViewModel Design Pattern | Microsoft Learn, accessed February 1, 2026, https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern
5. KUKA System Software 8.3, accessed February 1, 2026, http://www.wtech.com.tw/public/download/manual/kuka/krc2ed05/Operating%20

and%20Programming.pdf
6. C4 Color Palette, accessed February 1, 2026, https://www.color-hex.com/color-palette/3888
7. Orange grey color theme - Adobe Color, accessed February 1, 2026, https://color.adobe.com/t-arm-Kuka-2-color-theme-20455545/
8. WPF Touchscreen User Interface Guidelines - Stack Overflow, accessed February 1, 2026, https://stackoverflow.com/questions/1116342/wpf-touchscreen-user-interface-guidelines
9. Expert Programming KUKA System Software (KSS) Release 5.2, accessed February 1, 2026, https://icdn.tradew.com/file/201606/1569362/pdf/7033432.pdf
10. Edit Form | WPF Controls - DevExpress Documentation, accessed February 1, 2026, https://docs.devexpress.com/WPF/403491/controls-and-libraries/data-grid/data-editing-and-validation/modify-cell-values/edit-form
11. Helix Toolkit, accessed February 1, 2026, https://helix-toolkit.github.io/
12. HelixToolkit.Core.Wpf 2.27.3 - NuGet, accessed February 1, 2026, https://www.nuget.org/packages/HelixToolkit.Core.Wpf
13. How to set up a 3d model using HelixToolkit.Wpf.SharpDX? - Stack Overflow, accessed February 1, 2026, https://stackoverflow.com/questions/65766435/how-to-set-up-a-3d-model-using-helixtoolkit-wpf-sharpdx
14. Robot Simulator using Helix ToolKit WPF 3D - YouTube, accessed February 1, 2026, https://www.youtube.com/watch?v=g11Y4vOgXYo
15. Automotive Robot Controllers — UX Design Patterns & Benchmarking | by Creative Navy, accessed February 1, 2026, https://medium.com/@creativenavy/automotive-robot-controllers-ux-design-patterns-benchmarking-9bdd4f58d616