FACIAL MATCHING DISPLAY MIRROR

ECE 445 Senior Design Final Report

By

Keenan Peris
Krish Sahni
Connor Tan

Team 97
TA: Argyrios Gerogiannis
Professor: Yang Zhao
Project No. 97

May 2026

Abstract

Public science outreach exhibits often rely on static posters or non-interactive displays that fail to engage visitors and rarely communicate that science careers are accessible to people from a wide range of backgrounds. The Facial Matching Display Mirror is a touchless, camera-driven exhibit that addresses this gap. The system presents itself as a conventional mirror until a visitor approaches, at which point it transitions into an interactive display, prompts the user to select a STEM career area through a hover-based gesture interface, captures the user's face, and matches them against a local database of enrolled science and engineering professionals. The matched professional's profile, including name, role, organization, image, and biography, is then displayed behind a 70%-reflective two-way acrylic mirror.

The system is built on a distributed embedded architecture. A custom STM32F401 PCB was designed to act as the deterministic system controller, handling presence detection, system state transitions, peripheral control, and UART communication. A Raspberry Pi 4 with 8 GB of RAM serves as the perception worker, running MediaPipe-based hand tracking, InsightFace face embedding, SQLite database matching, and OpenCV-based UI rendering. Communication between the two processors is performed over a single-byte UART command protocol at 115200 baud. The display behind the two-way mirror is driven by HDMI from the Raspberry Pi, and a VL53L3CX time-of-flight sensor is used as the primary presence-detection element.

During development, brown-out behavior was observed on the custom PCB during normal operation, traced to insufficient transient capacity on the 3.3 V rail. Because there was no second board iteration prepared, the team substituted an Arduino Uno as a microcontroller stand-in for the final demonstration. A camera-based bounding-box presence-detection path was also added in addition to the time-of-flight sensor to minimize false positives. The final integrated prototype reliably executes the full interaction flow under demonstration conditions and meets the four high-level requirements defined at the start of the project.

Contents

1. Introduction

1.1 Problem and Motivation

Public museums, science exhibits, and STEM outreach programs face a recurring challenge: communicating complex science and engineering concepts in a way that engages a broad audience and helps non-experts feel that these fields are accessible to them. Static displays and informational panels are inexpensive but fail to hold attention, particularly for younger visitors who are accustomed to interactive digital experiences. The problem is sharper for emerging fields such as quantum information science, where the underlying ideas are unfamiliar and the public-facing scientists and engineers are not widely recognized.

This project was motivated by a partnership with the Illinois Quantum and Microelectronics Park (IQMP), which is interested in outreach exhibits that introduce visitors to careers in quantum science and engineering. The goal is to give visitors a personal, low-friction way to see themselves connected to a real working scientist or engineer, and to learn what that person does, where they work, and how they got there. A mirror is a particularly appropriate medium for this goal because it inherently invites the visitor to view themselves; turning that mirror into a display that introduces a matched STEM professional makes the connection concrete.

1.2 Solution Overview

The Facial Matching Display Mirror is an interactive exhibit that switches between a reflective mirror state and an active digital display. When no visitor is present, the system appears as a normal mirror. When a visitor approaches the interaction zone, the system activates and presents a touchless career-selection interface. The visitor selects a STEM career area by hovering an index-finger cursor over a button on screen for a fixed dwell period. The system then captures the visitor's face from a webcam, computes a face embedding, searches a local database of enrolled professionals filtered by the selected career area, and displays the closest matching professional's profile on the screen behind the mirror.

The design uses two coordinated processors. A custom STM32F401 PCB was developed to serve as the deterministic system controller, responsible for presence detection over I2C, finite-state-machine execution, peripheral control, and UART communication. A Raspberry Pi 4 with 8 GB of RAM runs the perception pipeline, including MediaPipe hand tracking, InsightFace face embedding, SQLite database queries, and OpenCV-based UI rendering, and drives the display over HDMI. The two processors communicate using a low-bandwidth single-byte command protocol over UART at 9600 baud, which keeps timing deterministic and avoids forcing image data onto the wire. Figure 1.1 shows the system block diagram.

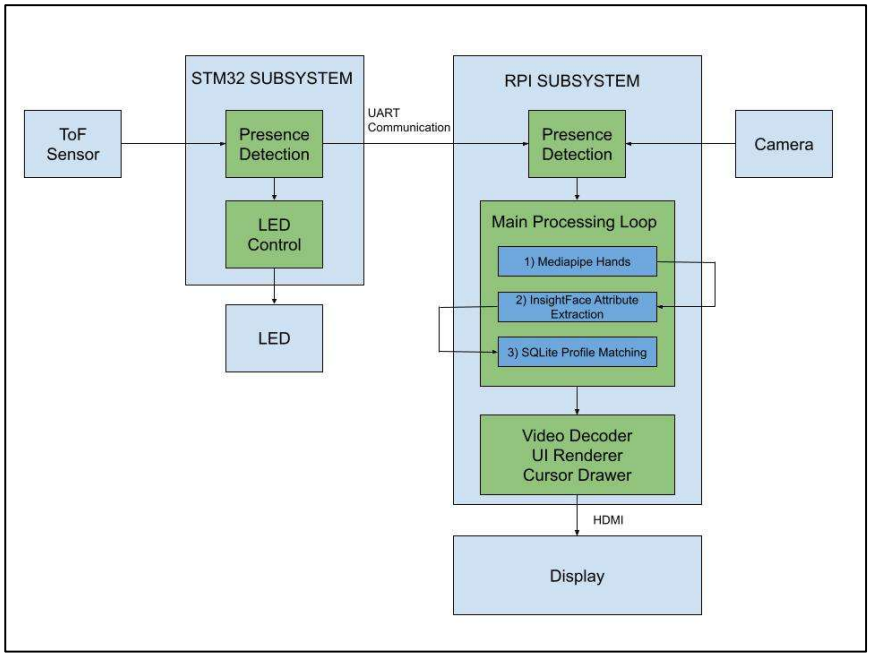

Figure 1.1: System block diagram showing the STM32 controller, Raspberry Pi 4 perception worker, and shared peripherals.

1.3 Visual Concept

The mirror illusion is created by a 70%-reflective, 30%-transmissive two-way acrylic film mounted in front of a 32-inch LED display. When the display is dark and ambient room light is moderate, reflected light dominates and the surface looks like a normal mirror. When the display drives bright, high-contrast graphics, the transmitted light dominates and the user sees the UI through the acrylic. An LED strip mounted on the bezel is controlled by GPIO and provides additional ambient lighting cues during the idle-to-active transition. Figure 1.2 illustrates the principle.

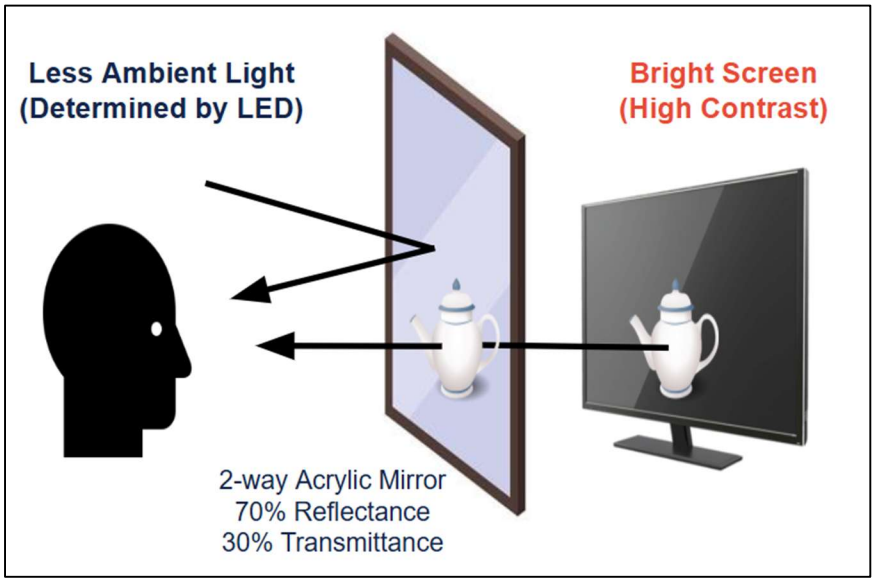

Figure 1.2: Two-way mirror principle. With LED off and the screen dark, the surface acts as a mirror; with the screen bright, UI elements are visible through the 70/30 film.

1.4 High-Level Requirements

To consider the project successful, the system must satisfy four high-level requirements:

1. The system must detect the presence of a user standing in front of the mirror within 3 seconds and activate the display.
2. The system must identify or match a user to a professional from a predefined database and display the associated profile information on the screen.
3. The mirror must function as a reflective surface when inactive and seamlessly transition to an interactive display when a user is detected.
4. The system must present readable visual information, including the matched professional's name, image, role, and short biography, while maintaining the mirror illusion.

1.5 Block-Level Changes During the Semester

Two functional-block changes were made during the semester. First, the original face-matching backend was based on OpenCV Haar classifiers and dlib embeddings. After the midpoint demonstration, this was replaced with InsightFace, which produces normalized recognition embeddings of higher quality and allowed the matching pipeline to use straightforward cosine similarity in place of the earlier ad-hoc comparison logic. Second, a camera-based bounding-box presence-detection path was added in addition to the time-of-flight (ToF) presence-detection path after the VL53L3CX showed false positives for presence detection. The fault was traced to how the ToF sensor worked. The two presence-detection paths run one after the other in the final system, and the first to report a confirmed presence event activates the system.

A third change was forced by hardware behavior rather than design intent. Brown-out behavior was observed on the custom STM32F401 PCB during normal operation, in which the 3.3 V rail momentarily dropped below the brown-out reset threshold under load, causing the microcontroller to reset. Because the schedule did not include a second PCB iteration, the team substituted an Arduino Uno as a microcontroller stand-in for the final demonstration. The UART command protocol was unchanged, so the substitution did not affect the rest of the system. The custom PCB design is presented as the intended hardware throughout this report; verification details for the brown-out behavior and the Arduino fallback are discussed in Chapter 3.

2. Design

2.1 Overall System Architecture

The Facial Matching Display Mirror is implemented as a distributed embedded system using two coordinated processors. The MCU is the deterministic system controller. It runs without an operating system in a 1 kHz main control loop, executing the system finite state machine, polling the time-of-flight sensor over I2C, controlling status LEDs and the bezel LED strip, and communicating with the Raspberry Pi over UART. The Raspberry Pi 4 runs Raspberry Pi OS and executes the perception pipeline: MediaPipe hand tracking, InsightFace face embedding, SQLite database queries, and OpenCV-based UI rendering. The Raspberry Pi drives the display over HDMI.

Three rationales drove this split. First, the perception tasks are computationally heavy and require Python libraries that depend on a Linux environment, while the control tasks require predictable, sub-millisecond response to peripheral events. Running both classes of tasks on the same processor would either delay the control loop or starve the perception pipeline of CPU. Second, the partition isolates faults: a Python crash on the Pi cannot lock up the state machine, and a UART glitch from the STM32 cannot corrupt the database.

Communication between the two processors uses a single-byte command protocol over UART at 9600 baud. UART was chosen over SPI and I2C. SPI was rejected because its master/slave model is poorly suited to the bidirectional, event-driven traffic in this system, and because the Pi's SPI peripheral could conflict with display output. I2C was rejected because the Linux user-space I2C-slave support is poor on the Pi side. UART is natively supported on both devices, requires only two signal lines, and matches the low-bandwidth nature of the actual traffic (cursor coordinates, selection events, presence events, and match notifications). Image data is never sent over the link.

2.2 Power Subsystem

The STM32 PCB receives 5 V from a USB Type-C receptacle configured as a downstream-facing port. The 5 V VBUS input passes through a Schottky diode for reverse polarity protection, then through a USBLC6-2SC6 ESD suppression IC on the data lines, before reaching the on-board low-dropout regulator. An AP2112K-3.3 LDO regulates 5 V down to 3.3 V to power the MCU, the VL53L3CX time-of-flight sensor, and the digital logic for the LED indicators. Figure 2.1 shows the regulated power chain.

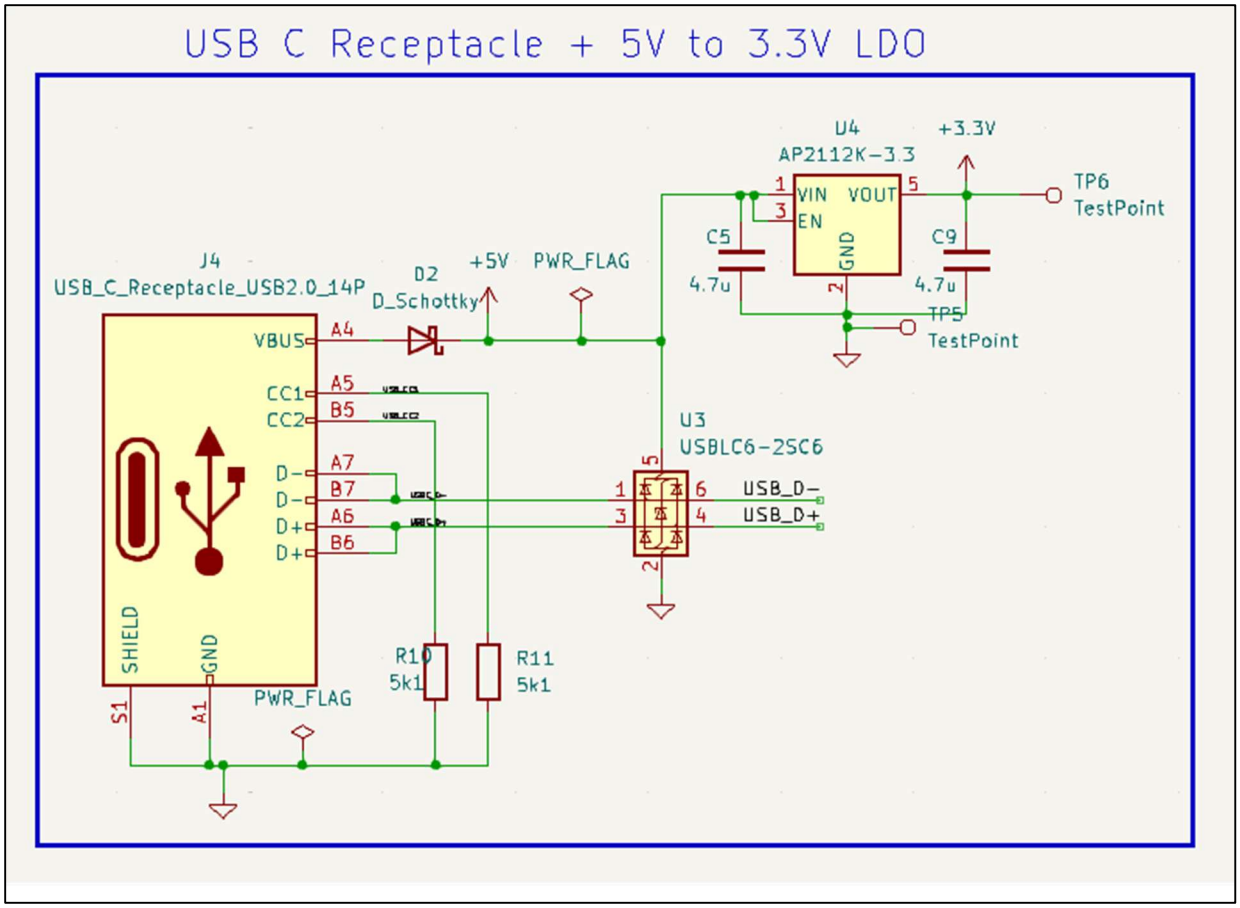

Figure 2.1: Power distribution chain. USB-C VBUS feeds a Schottky reverse-protection diode, ESD protection on the data pair, and the AP2112K-3.3 LDO. Local decoupling is distributed across the 3.3 V plane.

An LDO was selected over a switching regulator because the total steady-state load on the 3.3 V rail (STM32F401 plus ToF sensor plus LED logic) is well under 300 mA, which is comfortably inside the AP2112K's 600 mA capability.

Decoupling follows the recommendations in the STM32F401 datasheet [1]. A 4.7 µF bulk capacitor sits at the regulator output, and 100 nF ceramic capacitors are placed at every VDD pin of the STM32, oriented for the shortest possible return path to the ground plane. A 2.2 µF capacitor is connected to the VCAP pin to stabilize the STM32's internal core regulator. The LED strip on the bezel and the 32-inch display are powered from their own AC adapters and remain electrically isolated from the low-voltage control rail; only ground is shared between subsystems where signal references are required.

The brown-out detector inside the STM32 was enabled at the 2.9 V threshold to prevent flash corruption during undervoltage events. As discussed in Chapter 3, this detector did trigger during testing, which is what surfaced the brown-out behavior on the assembled board.

2.3 STM32F401 Embedded Control

The STM32F401 was selected as the system controller because it provides the peripherals needed by this project (hardware UART, hardware I2C, sufficient GPIO, and USB) at a price point and package size compatible with a single 4-layer PCB. It is also available on the Nucleo development board provided by the course, which gave the team a direct fallback if the custom board failed entirely. Figure 2.2 shows the STM32 core schematic, including decoupling, boot configuration, SWD header, and USB interface.

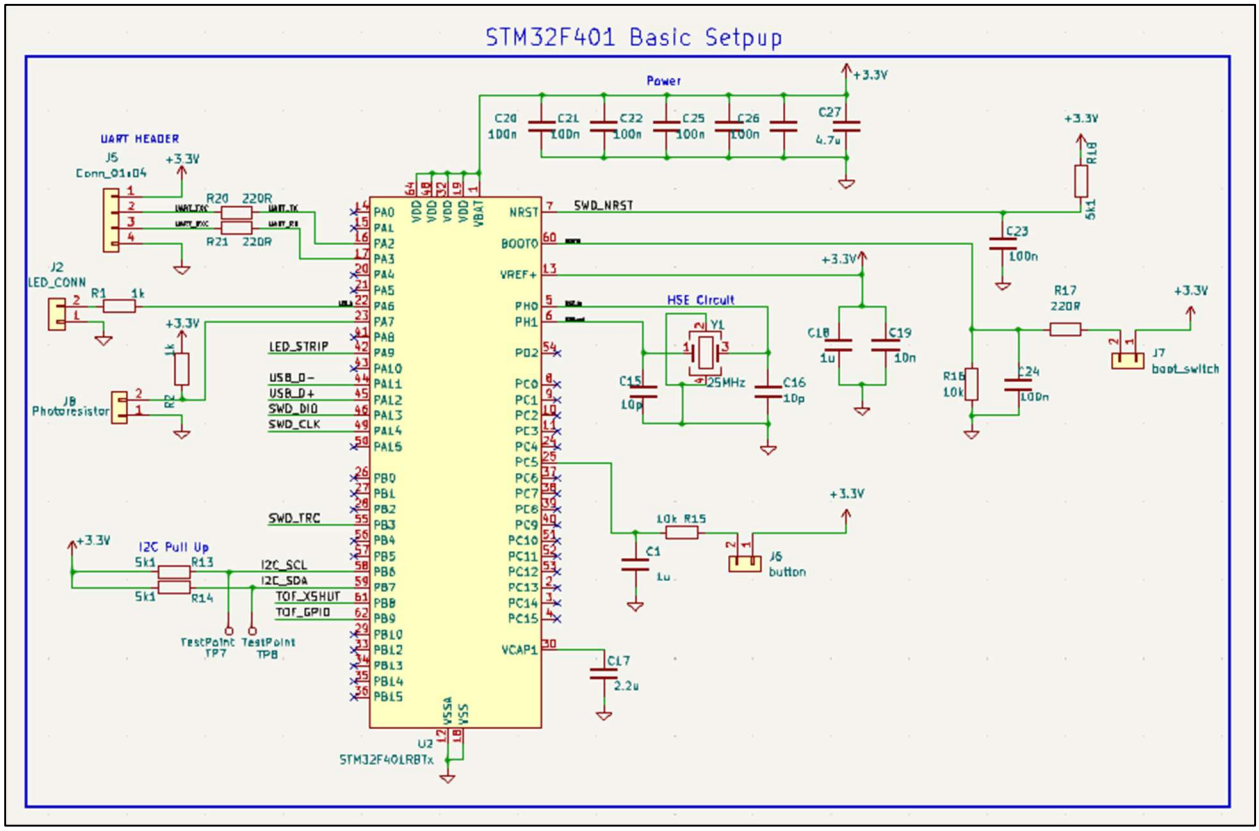

Figure 2.2: STM32F401 core schematic. Decoupling capacitors, NRST circuit, BOOT0 pull-down, SWD programming header, and USB OTG_FS pair are shown. The HSE crystal block is detailed in Figure 2.3.

2.3.1 Reset, Boot, and Programming

The NRST pin uses a 10 kΩ pull-up to 3.3 V and a 100 nF capacitor to ground for stable power-on reset behavior. A manual reset pushbutton is also provided for development. BOOT0 is held low through a 10 kΩ resistor so that the STM32 boots from internal flash by default during normal operation. An SWD header exposes SWDIO, SWCLK, ground, and 3.3 V, and is the primary programming and debug path.

As a backup, the STM32's USB_OTG_FS D+/D− pair is routed to the USB-C connector. If the SWD interface becomes unavailable during assembly or test, the STM32's built-in DFU bootloader can be activated by pulling BOOT0 high and connecting USB to a host running

STM32CubeProgrammer. The USB peripheral requires a precise 48 MHz clock derived from the PLL, so the USB DFU path depends on a working HSE crystal.

2.3.2 Finite State Machine

The STM32 firmware runs a deterministic finite state machine in a main loop operating at approximately 1 kHz. Five primary states are defined: Idle (Mirror Mode), UI Selection Mode, Face Validation Mode, Profile Display Mode, and Timeout-Return-to-Idle. State transitions are driven by validated presence detection from the ToF subsystem, command bytes received from the Raspberry Pi over UART, face-confidence threshold confirmation, and inactivity timers. If no presence is detected for 10 seconds, the system automatically returns to Idle to reduce processing and avoid leaving the display active when the visitor has walked away. Because the STM32 runs without a preemptive operating system, state transitions occur without scheduling jitter and respond to events in well under one millisecond. Figure 2.3 shows the state diagram.

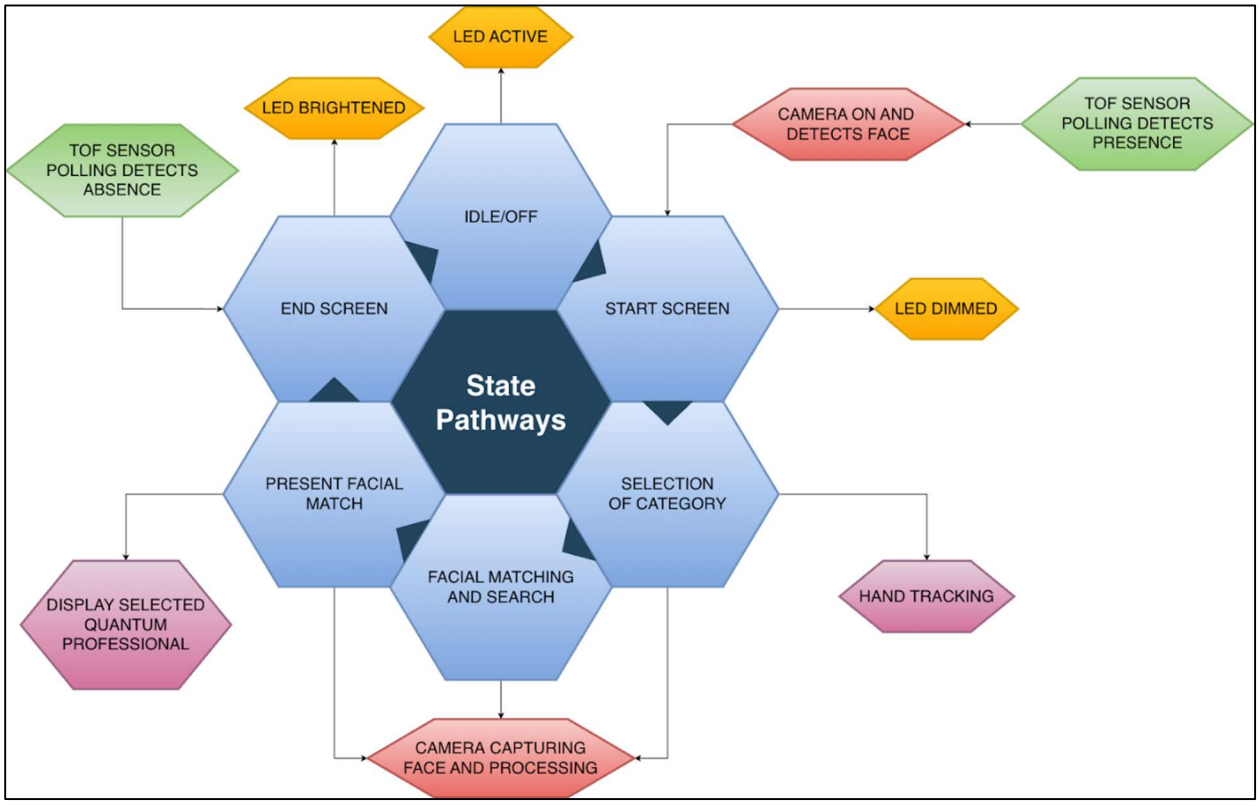

Figure 2.3: System finite state machine, showing Idle, UI Selection, Face Validation, Profile Display, and Timeout-Return-to-Idle states with their transition conditions.

2.4 Presence Detection

Presence detection is performed by a VL53L3CX time-of-flight sensor connected to the STM32 over I2C. The sensor emits short infrared light pulses and measures the round-trip time of the reflected pulse to compute distance using Equation (2.4), where d is the measured distance, c is the speed of light, and t is the round-trip time.

$$d = (c \times t) / 2 \quad (2.4)$$

The I2C bus uses 5.1 kΩ pull-up resistors to 3.3 V on both SDA and SCL, taken directly from the ToF sensor datasheet recommendation [4]. With 3.3 V supplies and 5.1 kΩ pull-ups, the pull-up current is approximately 0.65 mA, which is within the STM32's GPIO sink capability and produces adequate rise times at the 400 kHz I2C fast-mode clock. Additional GPIO lines are used for the sensor's XSHUT shutdown pin and INT interrupt output.

The STM32 polls the ToF sensor at approximately 15 Hz. The interaction zone is defined as a measured distance between 1.5 m and 3.5 m, which keeps the user inside the camera's framing window and at a comfortable conversational distance from the display. To reduce false activations from transient motion or measurement noise, the measured distance must remain inside this range continuously for 500 ms before presence is confirmed and the system transitions out of Idle. Total activation latency from the moment a user enters the zone to the activation of the UI is therefore approximately 505 ms, well inside the three-second high-level requirement.

2.5 Display and Mirror Enclosure

The display assembly consists of a 32-inch LED panel mounted behind three 12 × 24 inch acrylic panels that are coated with 70%-reflective, 30%-transmissive mirror film. The acrylic panels tile horizontally to cover the visible area of the display while leaving a small bezel around the edges for mechanical mounting. When the display drives near-black content, the small fraction of transmitted light is dominated by the 70%-reflective return path of ambient room light, and the surface looks like a mirror. When the display drives a bright, high-contrast UI, the transmitted light dominates the user's perception and the UI is visible through the acrylic. High-contrast graphics are used throughout the UI to maintain visibility under a range of ambient conditions.

Because the acrylic mounting hardware reduces the visually usable display area, the UI rendering code is parameterized to fit inside the visible region rather than the full panel resolution. Visible-width and visible-height ratios are computed at startup from the original screen dimensions and the measured visible region. These ratios feed directly into the layout configuration, which scales button size, margin, spacing, font scale, and progress-ring size proportionally. This avoids the alternative of hand-measuring and hard-coding UI element coordinates each time the bezel changes.

A short LED strip mounted along the bezel is controlled from a STM32 GPIO via a transistor driver. It provides ambient lighting cues during the idle-to-active and active-to-idle transitions, reinforcing the change in state for the visitor and softening the visual transition into the bright UI.

2.6 Camera and Perception Pipeline

Camera input is captured from a Logitech C920 USB webcam mounted behind the upper edge of the mirror and aimed forward into the interaction zone. The camera operates at 1280 × 720 resolution and 30 frames per second. Frames are flipped horizontally on capture so that the visitor's hand movement on screen matches the natural mirror direction. To reduce computational load,

frames are downscaled to 640 × 480 before further processing, which reduces pixel count by approximately 56 percent while preserving sufficient resolution for face detection and hand-landmark inference. End-to-end frame acquisition and preprocessing latency is approximately 20 to 30 milliseconds per frame under normal conditions on the Raspberry Pi 4.

2.6.1 Hand Tracking and Cursor

Hand tracking is implemented using MediaPipe's 21-landmark hand model. The HandTracker module wraps MediaPipe and exposes a single output: the normalized (x, y) coordinate of landmark 8, the index fingertip, in the range [0, 1], or None if no hand is detected in the current frame. The fingertip coordinate is mapped from camera space to display space using proportional scaling against the same visible-region ratios computed for the UI layout, so the cursor lines up with on-screen buttons regardless of bezel size.

2.6.2 Hover-to-Select Interaction

Selection is implemented by hover-and-dwell rather than by a click gesture. The user holds the cursor inside a button boundary for 1.5 seconds, during which a circular progress ring is drawn over the button to provide visual feedback. When the elapsed hover time crosses the dwell threshold, the button emits a selection event and the hover state is reset to prevent immediate re-selection. Hover-and-dwell was chosen because it is robust against the variability of MediaPipe's hand-pose detection, requires no assumptions about hand orientation, and is readable to first-time users without instructions. Figure 2.4 shows the three primary UI screens: career selection, matching in progress, and profile display.

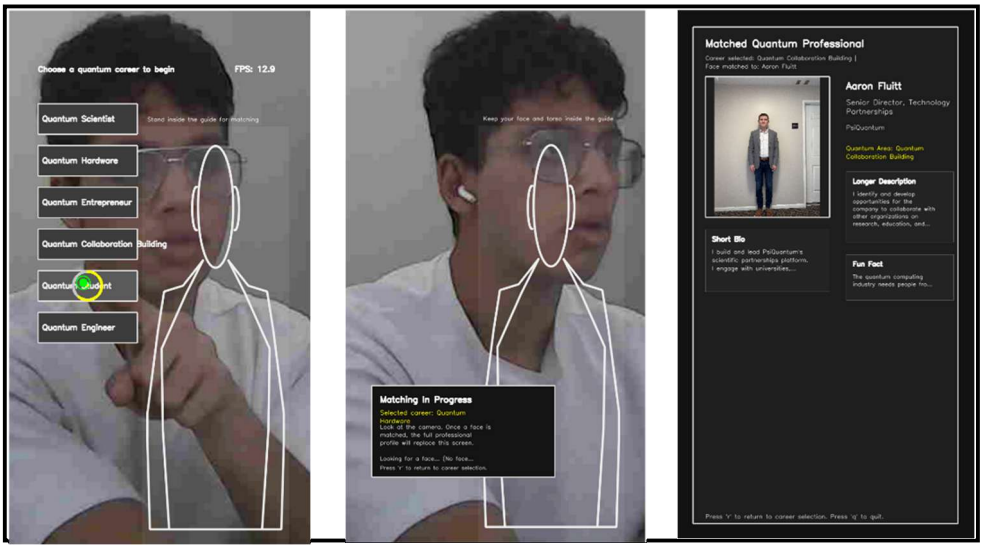

Figure 2.4: Three-screen UI flow rendered through the two-way mirror. Left: career selection with hover progress ring. Center: matching in progress with face-position guide outline. Right: matched professional profile with name, role, organization, and biography.

2.6.3 Confidence Gating Before Embedding

Face detection is performed using OpenCV-based classifiers running on the Raspberry Pi. Before the system commits CPU time to running InsightFace embedding, a composite confidence score is computed from the face detection result, as defined in Equation (2.5):

$$C = 0.4 \cdot P_d + 0.3 \cdot (A_{face} / A_{frame}) + 0.3 \cdot V_{laplacian} \quad (2.5)$$

Here P_d is the face detector's reported probability, A_face is the bounding-box area of the detected face, A_frame is the full frame area, and V_laplacian is a Laplacian-variance sharpness metric computed on the face region. A confidence score above 0.65 is required before the frame is passed to the embedding stage. This gate filters out partial faces, frames captured during fast head motion, and frames where the user is too close to the camera or too far from it. End-to-end perception latency from frame capture through confidence evaluation, embedding, and database query stays below approximately 65 milliseconds, which is fast enough that the matching state feels responsive to the user.

2.7 Facial Matching and Database

2.7.1 Why Embeddings

The matching subsystem represents each face as a numerical embedding rather than comparing raw images. An embedding is a fixed-length vector that captures facial features in a form that can be compared mathematically, and it is the standard input to modern face-recognition pipelines. The system uses InsightFace's buffalo_sc model pack, running on the CPU execution provider, to produce normalized recognition embeddings for both enrolled professionals and live visitors.

2.7.2 Database Schema

Professional data is stored in a local SQLite database. SQLite was chosen because it runs locally on the Pi, requires no network connection or external server, and supports the privacy goal of keeping all face-matching state on the device. The schema defines five tables: professionals, profile_tags, interaction_logs, face_embeddings, and demo_profile_links. The professionals table stores each enrolled professional's name, title, organization, quantum or career area, short biography, longer description, image path, fun fact, optional video URL, and creation timestamp. The profile_tags table assigns career-area tags to professionals and supports many-to-many membership. The face_embeddings table stores the precomputed embedding for each professional under a (professional_id, model_name) uniqueness constraint, so embeddings from different models can coexist without overwriting each other. Embeddings are stored as JSON-encoded float lists. The interaction_logs table records demonstration events for later inspection.

2.7.3 Enrollment

Professional enrollment runs offline before the demonstration. The enrollment script iterates over all rows in the professionals table, locates each professional's headshot at the path stored in image_path, runs the InsightFace embedder on the image, and writes the resulting embedding into face_embeddings using upsert_face_embedding. Once enrollment completes, the live system

never recomputes professional embeddings, only visitor embeddings, which keeps the per-frame matching cost low. During development, an early bug in this pipeline was found: image paths stored in the database did not match the actual locations of files on disk, so several enrollments silently failed. A diagnostic script was added that lists professionals whose image_path does not resolve, which made the path mismatch easy to detect and fix.

2.7.4 Matching Algorithm

Face matching compares the visitor's embedding against the enrolled professional embeddings using cosine similarity, defined in Equation (2.6):

$$sim(a, b) = (a \cdot b) / (\|a\| \cdot \|b\|) \quad (2.6)$$

Cosine similarity measures the angle between two vectors rather than their magnitude difference, which is the appropriate comparison for normalized embedding vectors. The find_best_database_matches function loads all enrolled embeddings for the active model name, optionally filters them by an allowed set of professional IDs (which is how the user's career-area selection narrows the search before matching), ranks the remaining candidates by similarity, and returns the top-k matches together with their full profile rows in a single batch SQL query rather than the N+1 query loop the earlier implementation used.

2.7.5 Result Framing

Because the system uses a small enrolled database and a CPU-only embedder, the result is framed in the user-facing copy as a 'lookalike' or similarity-based match rather than as identity recognition. This framing is also consistent with the privacy posture: the system never claims to know who the visitor is, only which enrolled professional their face is closest to under the embedding model.

2.8 UART Communication Protocol

The UART link between the Raspberry Pi and the STM32 operates at 9600 baud. The Pi's TX pin connects to the MCU UART_RX and the Pi's RX pin connects to the MCU's UART_TX. The protocol uses single-byte commands rather than a framed packet structure, because the actual traffic in this system is a small set of discrete events (presence change, cursor update, selection event, match notification, mode change), and a byte-per-event scheme has the lowest possible serialization overhead.

2.9 PCB Layout

Component placement on the custom PCB followed standard embedded layout practice. Decoupling capacitors were placed as close as possible to their respective VDD pins, with short, direct traces to the ground plane. The HSE crystal and its two 10 pF load capacitors were placed immediately adjacent to OSC_IN and OSC_OUT, with a ground guard ring to minimize stray capacitance and crosstalk into the oscillator network. The USB D+/D− pair was routed as a controlled-impedance differential pair with matched trace lengths. The I2C and UART connectors were placed on the board edge for easy cable access in the assembled enclosure, and the SWD

header was placed in an open area for programmer clip-on access during development. The full schematic is included intact in Appendix B; Figure 2.5 shows the routed board.

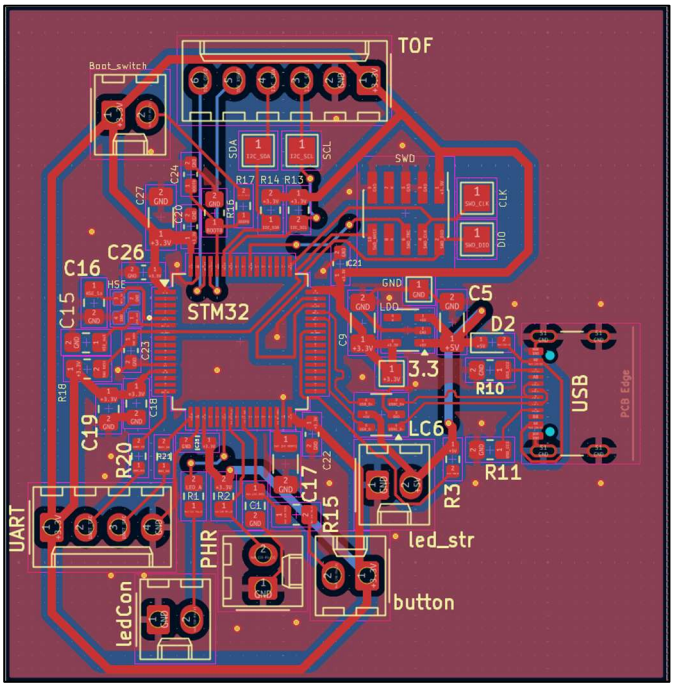

Figure 2.5: Routed PCB layout. Routing was performed after placement was finalized.

3. Verification

This chapter summarizes the verification activities performed for each subsystem. Per the report guidelines, only the higher-level results and the requirements that failed or required follow-up are discussed in detail in the main text. The full Requirement and Verification table appears in Appendix A; readers interested in the verification status of a specific low-level requirement are directed there.

3.1 Power and STM32 Bring-up

Initial bench verification of the assembled custom PCB was straightforward. The 3.3 V rail was measured at the test point with a multimeter under no load and read within tolerance of the AP2112K specification. With the STM32 not yet flashed, the rail also held under the standby current of the populated peripherals.

Brown-out behavior surfaced once the STM32 firmware began driving GPIO transitions. With the Arduino-stand-in firmware temporarily flashed onto the STM32 to drive UART and the LED outputs simultaneously, the STM32's brown-out reset triggered intermittently and the microcontroller reset. Probing the 3.3 V rail with an oscilloscope at the moment of reset showed a transient dip below the 2.9 V brown-out threshold coincident with the LED-on edge. The cause was traced to insufficient transient capacity on the 3.3 V rail: the bulk capacitance at the LDO output and the local decoupling were sized for steady-state operation but did not absorb the load step from the LED switching. Increasing the bulk capacitance on the 3.3 V output of the LDO is the obvious fix, and would have been the first change in a second board iteration. Because the project schedule did not include a second PCB spin, the team chose to use an Arduino Uno as a microcontroller stand-in for the demonstration. The Uno hosts the same UART command protocol as the STM32 firmware, so the rest of the system, including the Pi-side software, was unaffected by the substitution. The brown-out finding and the lack of a planned second iteration are reflected as 'lessons learned' in Section 5.5.

SWD flashing was verified using an ST-Link programmer connected to the on-board SWD header. A blink-LED firmware image flashed and ran successfully, which confirms that the STM32, the SWD circuit, and the GPIO test pin are all functional. The USB DFU programming path was not exercised because SWD was sufficient for development; the DFU path remains in the design as a backup.

3.2 Presence Detection

ToF presence detection was tested at three positions in the interaction zone (1.7 m, 2.5 m, and 3.3 m) and at three positions outside the zone (1.0 m, 4.0 m, and a position with no person in front of the sensor). Inside the zone, the 500 ms hold filter consistently confirmed presence and triggered a state transition; outside the zone, no transitions occurred during sustained five-minute baseline runs.

3.3 UART Integration

End-to-end UART integration was verified before the custom PCB was available. The Raspberry Pi software, running on a development laptop in this test, was connected over USB-serial to an Arduino Uno that had been programmed to receive the project's command bytes (0x01, 0x02, 0x03) and toggle a status LED in response. With the hand-tracking UI running on the laptop, hovering an index fingertip over each on-screen button for the 1.5-second dwell period produced the expected byte transmission, the Arduino's LED state changed correctly, and the Arduino's acknowledgment string was received and printed at the laptop terminal. This test confirmed that the MediaPipe fingertip detection, the cursor smoothing and dwell logic, the byte-level UART transmission and reception, and the round-trip acknowledgment all worked together as designed. Figure 3.1 shows a representative terminal log.

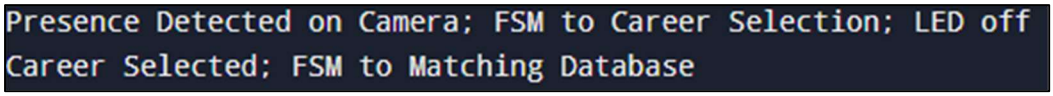

Figure 3.1: Terminal log captured during UART integration testing. Each hover-and-dwell selection produces a single byte transmission and a corresponding acknowledgment from the Arduino stand-in.

After the brown-out finding, the Arduino Uno used in this earlier test was promoted from 'integration stand-in' to 'demonstration controller' for the final demo. No protocol changes were required.

3.4 User Interface and Hand Tracking

The user interface was tested both on a development laptop and on the Raspberry Pi 4. The laptop ran the UI at approximately 18 frames per second, and the Pi 4 ran it at approximately 8-12 frames per second. Although the Pi's frame rate is lower, the interaction is hover-and-dwell with a 1.5-second threshold, so the user does not perceive the lower frame rate as a usability problem. Hover selections fire reliably at both frame rates, the circular progress ring is smooth and predictable, and short, accidental cursor motion across a button does not produce a selection. The cursor-linger timer of one second eliminated the previously observed ghost-cursor behavior in which brief MediaPipe drop-outs caused the cursor to disappear and reappear.

3.5 Database and Matching

Database initialization, professional population, and tag assignment were verified by running the population script against a fresh database and inspecting the resulting rows with the SQLite shell. All five tables were created with the expected columns, and duplicate-name handling correctly prevented re-inserting the same professional when the population script was run twice.

Face-embedding enrollment was verified using the diagnostic script that lists professionals whose stored image_path does not resolve on disk. After the path-mismatch fix described in Section 2.7.3, this script returned an empty list, and the enrollment run successfully populated face_embeddings rows for every enrolled professional under the InsightFace buffalo_sc model.

Matching was verified using the still-image query tool (query_image_demo.py). For each test image, the tool computes the query embedding, runs find_best_database_matches, and prints the ranked top-k professional names with their similarity scores. The relative ranking of matches was stable across repeated runs of the same query image, as expected for a deterministic embedding pipeline. Performance work resolved two issues during this phase: the original implementation issued one SQL query per candidate during the rank step, which produced an N+1 query pattern visible as noticeable per-query latency, and the original render loop called cv2.imread once per frame for the active professional headshot, which caused visible UI hitches. The N+1 pattern was replaced with a single batch fetch joined to the ranked results, and the render-loop image reads were replaced with a module-level _image_cache. After both fixes, the matching state runs without perceptible UI hitches.

3.6 Full System Integration

The complete interaction flow was verified end-to-end using the integrated demo. The system begins in a blank waiting state. When either the Arduino sends a PRESENCE message over UART or the Raspberry Pi's camera-bbox path detects a confirmed presence, the system opens the camera and enters the introduction state. The intro screen displays a short instruction with a countdown, and on completion the system transitions into career selection. The hand cursor selects a career area through the standard hover-and-dwell mechanism, after which the system filters the candidate set of professionals by the selected career area and enters the matching state. A guide outline is drawn on the display to help the visitor center their face inside the matching region. The cropped region of interest is submitted to a background embedding worker thread once every 1.5 seconds; if the worker is still busy from the previous job, the new frame is dropped rather than queued, which keeps the UI responsive. When the worker returns a valid match, the system transitions to the profile screen, displays the matched professional's name, title, organization, image, biography, and fun fact, and sends a MATCH message over UART so the controller advances its state. After a fixed display interval the system returns to idle.

This complete flow was demonstrated successfully under indoor lighting on a development bench and during the final demonstration day. Two limitations are worth noting. First, the matching state currently fires its embedding worker every 1.5 seconds; under more variable lighting or with a partially obscured face, the visitor sometimes sees a short delay between entering the matching state and seeing the profile, longer than the design target. Second, although the Requirement and Verification table notes 4.4 (the system avoids rapid accidental selections) as fully met, the team observed that very fast hand sweeps across the cursor area can occasionally produce a selection if the hand path crosses a button at exactly dwell speed; this is an edge case and was not seen during the demonstration.

4. Costs

4.1 Labor

Labor cost is computed using the formula required by the course guidelines: ideal hourly salary, multiplied by actual hours worked, multiplied by 2.5. An entry-level salary for an Electrical or Computer Engineering graduate from the University of Illinois of approximately $85,000 per year corresponds to roughly $40 per hour assuming 40 hours per week over 52 weeks. Each of the three team members contributed approximately 120 hours over the semester.

Per partner: $40/hr × 120 hr × 2.5 = $12,000. Across all three partners: 3 × $12,000 = $36,000. Although this labor cost is not part of the prototype budget, it reflects the engineering value of the project.

4.2 Parts

Table 4.1 lists the major purchased components used in the prototype. Items marked 'Owned' had no out-of-pocket cost for this build. The custom PCB itself was fabricated under a small lot run and is reflected in the discrete component lines. The Logitech C920 webcam and Arduino Uno (used as the demonstration controller after the brown-out finding) were owned by team members.

Item	Vendor	Qty	Unit ($)	Total ($)
Raspberry Pi 4 Model B (8 GB)	Digi-Key	1	115.00	75.00
VL53L3CX ToF eval board	Digi-Key	1	8.02	8.02
32-inch LED display (used)	Provided by Professor Kwiat	1	0	0
70/30 acrylic mirror film + acrylic panels	Provided by Professor Kwiat	1	0	0
LED bezel strip	Amazon	1	9.99	9.99
AP2112K-3.3 LDO regulator	Digi-Key	5	0.21	1.05
USBLC6-2SC6 ESD protection	Digi-Key	5	0.36	1.80
USB-C receptacle	Digi-Key	3	0.71	2.13
25 MHz crystal (Kyocera CX2016SA)	Digi-Key	3	0.42	1.26
10-pin SMD header	Digi-Key	2	2.42	4.84
TVS diode (reverse-protection)	Digi-Key	5	0.33	1.65
Arduino Uno (demo stand-in)	Owned	1	0.00	0.00
Logitech C920 webcam	Owned	1	0.00	0.00
Wiring, connectors, mounting hardware	Various	—	—	25.00
Parts subtotal				**130.74**

Table 4.1: Prototype parts cost.

4.3 Grand Total

Combining estimated labor at $36,000 and prototype parts at $130.74, the full engineering value of the project is approximately $36,200. The hardware prototype cost remains comfortably under the $150 course budget. The largest single hardware cost is the Raspberry Pi 4 4GB, which was justified by the requirement to run InsightFace inference at acceptable latency on a CPU-only target.

5. Conclusion

5.1 Accomplishments

The Facial Matching Display Mirror successfully demonstrated a complete end-to-end interactive STEM outreach exhibit. Presence detection, touchless career selection through hover-and-dwell, face capture, InsightFace embedding, category-filtered cosine matching against a local SQLite database, and final profile display were all integrated and exercised on the demonstration day. The custom STM32F401 PCB was designed, fabricated, and bench-verified for power and basic firmware execution, even though the live demo ultimately used an Arduino Uno as a microcontroller stand-in. The UART command protocol between the perception worker and the system controller was validated end-to-end, both before the PCB was available and again on the integrated system. Six of the seven functional block requirements established at the start of the project were fully met, and the seventh (the MCU control block) was met in partial form using the Arduino stand-in.

5.2 Uncertainties and Limitations

Several limitations remain. The PCB brown-out behavior was identified but not fixed, and the second-iteration board that would have addressed it was not part of the original schedule. The Raspberry Pi 4 frame rate (~10 fps) is acceptable for hover-and-dwell interaction but not for any future feature that depends on faster gesture response. The enrolled professional database is small (under ten entries at demonstration time), so the displayed match should be interpreted as an illustrative lookalike rather than a statistically meaningful comparison. Finally, the InsightFace buffalo_sc model used for embedding has known accuracy variations across demographic groups [9], which means the system would require empirical evaluation across a broader user population before any public deployment.

5.3 Ethical Considerations

This project involves camera-based image capture and biometric similarity in a public-facing setting, which raises three categories of ethical considerations under the IEEE Code of Ethics [10].

On privacy: all face processing occurs locally on the Raspberry Pi 4. Visitor frames are passed into the embedding worker, the resulting embedding is used to query the database, and the frame is then released. No visitor images are written to disk, and no embeddings derived from visitors are stored. The database contains only enrolled professional images, professional metadata, and precomputed professional embeddings. Any deployed version of this system must include visible signage explaining that a camera is in use, what the system does with the captured image, and that no visitor data is retained. A visible camera-active indicator is included in the UI for the same reason. Any deployment in a public space should also include explicit opt-in before face capture begins.

On fairness and representation: face recognition systems are known to perform unevenly across demographic groups, and the embedding model used here is no exception [9]. Two mitigations are applied at the design level. First, the matching result is presented as a similarity-based 'lookalike,' not as identity recognition or as any judgment about the visitor's traits. Second, the enrolled database is intended to draw from a diverse set of professionals so that whatever match the system surfaces is a positive, accessible role model. A deployed version of this system should be evaluated empirically across skin tones, ages, gender presentations, lighting conditions, and camera angles before being made public, and the database should be sized large enough that any visitor receives a reasonable match.

On safety and standards: the entire control system runs on regulated low-voltage DC. All components operate inside their manufacturer-rated voltage and current envelopes. The display and acrylic mirror panels are mounted in a reinforced enclosure to prevent tipping, the acrylic edges are smoothed to reduce injury risk, and cable strain relief is provided at all connection points.

5.4 Broader Impacts

From a societal perspective, the system supports STEM outreach by giving visitors a personal, low-friction way to see themselves connected to a real working professional. From an economic perspective, the system is built from low-cost, commercially available components and can be replicated in schools, museums, and community-organization environments without significant infrastructure investment. From an environmental perspective, the system uses energy-efficient electronics, reduces reliance on printed materials, and processes everything locally without recurring cloud or network infrastructure.

5.5 Future Work

The most important short-term work is a second PCB iteration that addresses the brown-out behavior on the 3.3 V rail. The most plausible fix is increasing the bulk capacitance at the LDO output and adding additional local decoupling near the load points where transients were observed. Beyond hardware, two software directions stand out. First, expanding the enrolled professional database is the highest-leverage change for match quality and would benefit from a Google-Form-to-database pipeline so that professionals can be added without manually editing the population script. Second, optimizing the perception pipeline (for example, by exporting the InsightFace model to ONNX Runtime with appropriate threading) is the most plausible path to bringing the Pi 4 frame rate into the 15-fps range, which would expand the set of possible interaction styles. In terms of the deployment posture, an explicit on-screen consent step before face capture should be added before any public installation, and the consent copy should match whatever signage is used at the physical exhibit.

References

[1] STM32F401xB/STM32F401xC Datasheet, STMicroelectronics, 2023. Available: https://www.st.com/resource/en/datasheet/stm32f401cb.pdf

[2] USBLC6-2SC6 Datasheet, STMicroelectronics, 2014. Available: https://www.st.com/resource/en/datasheet/usblc6-2sc6.pdf

[3] CX2016SA Crystal Unit Datasheet, Kyocera AVX, 2025. Available: https://www.digikey.com/en/products/detail/kyocera-avx/CX2016SA25000D0FPLG1/16820398

[4] VL53L3CX Time-of-Flight Ranging Sensor Datasheet, STMicroelectronics, 2022. Available: https://www.st.com/resource/en/datasheet/vl53l3cx.pdf

[5] AP2112 600 mA CMOS LDO Regulator Datasheet, Diodes Incorporated, 2022. Available: https://www.diodes.com/assets/Datasheets/AP2112.pdf

[6] Raspberry Pi 4 Model B Datasheet, Raspberry Pi Ltd., 2023. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[7] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," Google Research, Tech. Rep., 2019. Available: https://mediapipe.dev

[8] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4690–4699.

[9] P. Grother, M. Ngan, and K. Hanaoka, "Face Recognition Vendor Test (FRVT) Part 3: Demographic Effects," NIST Interagency Report 8280, December 2019.

[10] IEEE Code of Ethics, IEEE, 2020. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[11] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 2nd ed. Sebastopol, CA: O'Reilly Media, 2016.

Appendix A Requirement and Verification Table

Table A.1 lists each system requirement, the verification procedure used to test it, and its final verification status. “Y” indicates the requirement was fully met, “N” indicates it was not, and “Partial” indicates that the requirement was met to the extent supportable by the tested configuration but not at the full quantitative target.

Requirement	Verification	Status
1a. The system shall detect a user automatically without manual activation.	Place a user within the 1.5–3.5 m interaction zone of the VL53L3CX and verify that the system transitions from idle to active without a button press.	Y
1b. The system shall activate consistently for multiple users.	Test the system with at least five users of varying heights and positions; record activation success rate.	Y
2a. The mirror shall transition from idle to digital content when a user is detected.	Observe the display during activation and verify that the screen changes from idle to the introduction or career-selection interface.	Y
2b. Displayed content shall be legible and aligned within the visible mirror area.	Inspect the UI on the final display and verify that buttons, text, guide outline, profile image, and biographical fields all remain inside the visible region computed from the visible-area ratios.	Y
3a. The system shall display a professional profile after a user selects a career and completes facial matching.	Select each available career category, perform the matching process, and verify that a valid profile (name, title, organization, image, biography) is shown.	Y
3b. The system shall restrict matching candidates to the selected career category.	Select a career category and verify that find_best_database_matches only compares the visitor embedding against professional IDs associated with that category.	Y
3c. The system shall avoid random profile activation before a valid match is returned.	Run the system without a face in the matching region and verify that the profile screen does not appear until the embedding worker returns a valid match.	Y
3d. The facial matching subsystem shall achieve greater than 80% reliability across repeated trials.	Run repeated matching trials across multiple test images and compute the percentage of trials in which the expected or acceptable match is returned. Limited by enrolled-database size and demographic coverage of the embedding model.	Partial (High reliability achieved but with a limited database size; must test on a bigger database)
4a. The system shall activate the user interface within 3 seconds of presence detection.	Use a timer to measure the delay between presence detection and the appearance of the active interface; repeat across trials and compare against the 3 s target. Measured activation latency ≈ 505 ms.	Y
4b. The user interface shall remain usable on the Raspberry Pi 4.	Run the final UI on the Pi 4 and observe whether the cursor, hover progress ring, and screen transitions remain responsive. Measured Pi-side frame rate ≈ 10 fps; sufficient for hover-and-dwell interaction.	Y

Requirement	Verification	Status
4c. The hover-selection system shall reliably trigger a selection after the dwell period.	Hover the cursor over each career button for the 1.5 s dwell time and verify that the corresponding career is selected.	Y
4d. The UI shall avoid rapid accidental selections.	Move the cursor briefly across buttons and verify that no selection occurs unless the cursor remains over a button for the full dwell time.	Y
5a. The system shall not store visitor face images.	Confirm that live visitor frames are processed in memory for embedding generation and are not written to the database or saved to disk.	Y
5b. The system shall store only enrolled professional images, metadata, and embeddings.	Inspect the SQLite schema and stored records to verify that the database contains only professional profiles, profile tags, interaction logs, and professional face embeddings.	Y
6a. The 3.3 V rail shall remain within tolerance under steady-state load.	Probe the 3.3 V test point with a multimeter under steady-state load and verify that the voltage remains within AP2112K-3.3 specification.	Y
6b. The 3.3 V rail shall remain within tolerance under transient load.	Probe the 3.3 V test point with an oscilloscope during GPIO/LED switching activity and verify that the voltage does not dip below the 2.9 V brown-out threshold.	N (brown-out observed; addressed by Arduino stand-in for demonstration)
6c. The STM32 shall be programmable via SWD.	Connect an ST-Link programmer to the on-board SWD header and successfully flash a blink-LED firmware image.	Y
6d. UART communication shall transmit and receive bytes at 9600 baud, without corruption.	Connect Pi to MCU (STM32 or Arduino stand-in), send the project's command byte set in both directions, and verify that bytes are received correctly with no protocol errors.	Y

Table A.1: System requirements and verifications.

Appendix B Full Schematic

Figure B.1 shows the complete schematic of the custom STM32F401 PCB. Sectioned views appear in Chapter 2. Insert the full schematic image below.

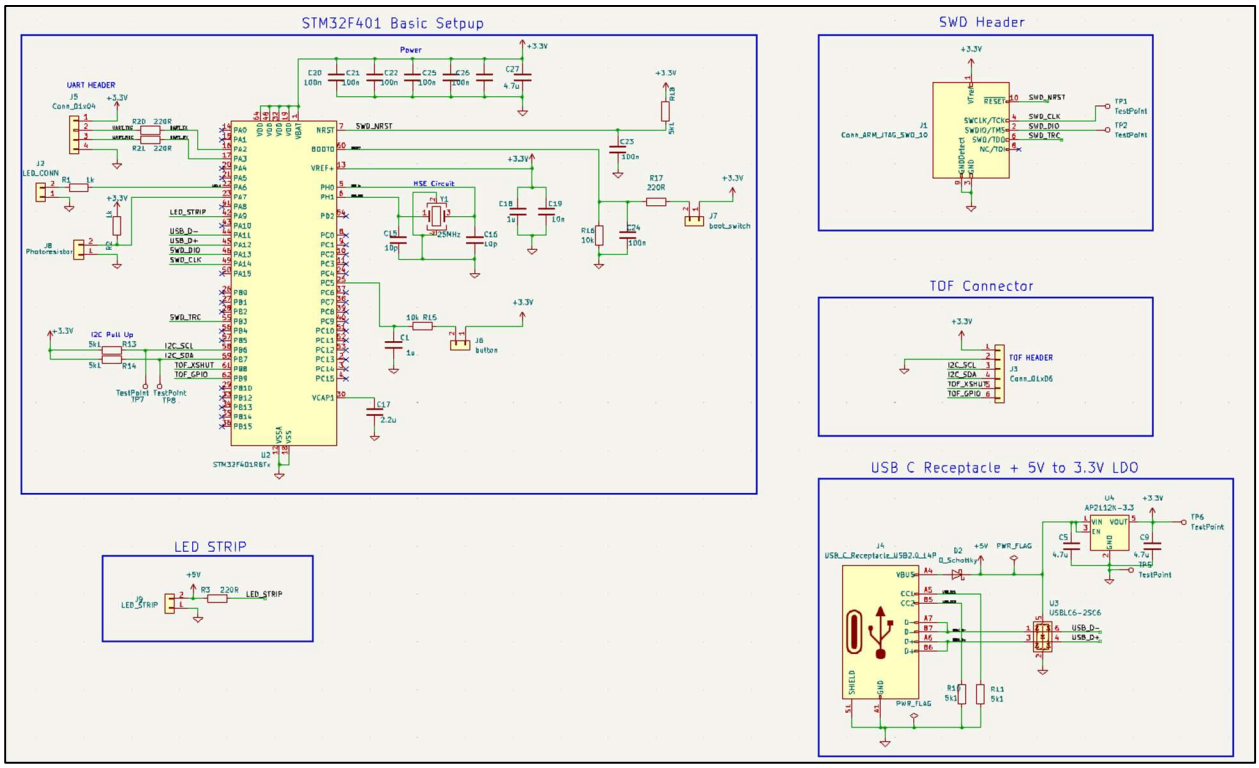

Figure B.1: Complete schematic of the custom STM32F401 PCB. Sectioned views are in Figures 2.1, 2.2, and 2.3.