# OpenFPT

## Open-Source Flying Probe Tester for PCB Electrical Evaluation

Comprehensive Technical Documentation Report

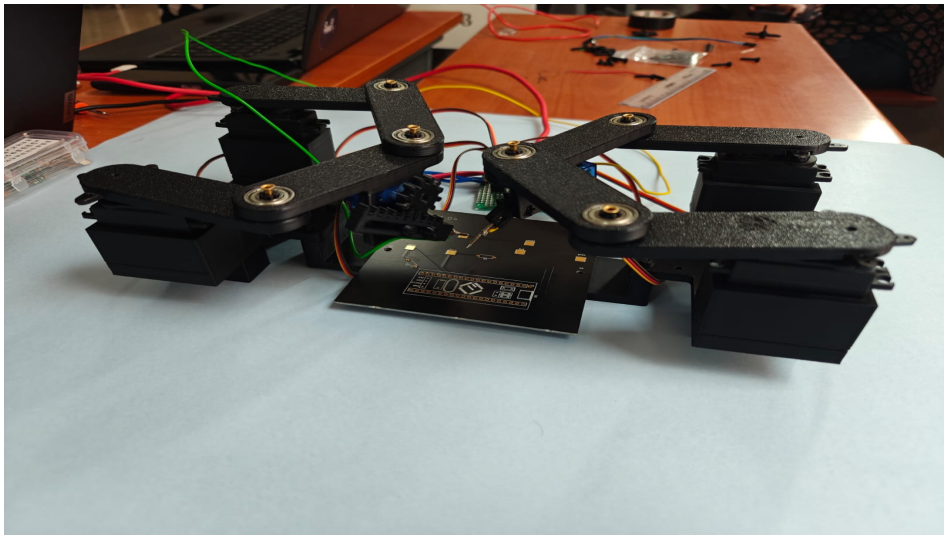

*Figure 0.1: OpenFPT Dual-Arm Prototype — Front View with PCB Under Test*

| | |
|---|---|
| **Project Title** | OpenFPT — Open-Source Flying Probe Tester |
| **Category** | Embedded Systems / Robotics / PCB Test Automation |
| **Platform** | ESP32-C6 (RISC-V) with ESP-IDF 5.5.2 |
| **Domain** | Automated PCB Electrical Testing & AI-Assisted Diagnostics |
| **Date** | April 2026 |

### Team Contact Details



| | |
|---|---|
| **Team Name** | OpenFPT |
| **Team Member 1** | Mithilesh — Phone: +91-8519962257 — Email: mithileshguptak@gmail.com |
| **Team Member 2** | Jishnu — Phone: +91-8125426744 — Email: jishnurudraraju@gmail.com |
| **Team Member 3** | Abhiram — Phone: +91-6281619764 — Email: abhiyanampally@gmail.com |

# Table of Contents

# 1. Title of the Project

**OpenFPT: An Open-Source Dual-Arm Flying Probe Tester for Automated PCB Electrical Evaluation with AI-Assisted Test Plan Generation and Fault Diagnosis**

OpenFPT is a dual-arm robotic probing platform designed for automated PCB electrical evaluation. The system employs two planar 2-DOF robotic arms based on 5-bar parallel linkage mechanisms, controlled by an ESP32-C6 microcontroller. Each arm positions a metallic probe over target test points on a PCB board. After positioning, dedicated probe actuation servos lower the probes to make physical contact with the board, and the system performs electrical measurements including continuity testing, isolation checks, and signal path verification. The platform integrates an AI-powered host software stack that automatically generates test plans from KiCad design files and provides LLM-based fault diagnosis from measurement results.

# 2. Problem Statement

Printed Circuit Board (PCB) testing is a critical step in electronics manufacturing and prototyping workflows. Before a PCB can be assembled and deployed, its electrical integrity must be verified: traces must conduct as designed, no unintended shorts exist, and key signal paths are correctly routed. In professional manufacturing environments, this task is performed by dedicated Flying Probe Testers (FPTs) — machines that use motorized probes to touch down on specific test points and measure electrical parameters.

However, commercial Flying Probe Testers present several significant barriers for smaller organizations, educational institutions, and hobbyist PCB designers:

- **Prohibitive Cost:** Commercial FPTs typically cost between USD 50,000 and USD 500,000, placing them well beyond the reach of small labs, startups, and educational programs.
- **Closed Architecture:** Existing systems are proprietary, with locked firmware, non-standard communication interfaces, and vendor-dependent test plan formats that prevent customization or academic study.
- **Manual Test Plan Creation:** Even when testing hardware is available, creating comprehensive test plans requires expert knowledge of the PCB layout, net assignments, and electrical characteristics — a tedious and error-prone manual process.
- **No Open-Source Alternative:** As of 2026, no open-source flying probe test platform exists that provides end-to-end coverage from mechanical design through firmware to test execution and diagnosis.
- **Limited Integration with Modern Design Tools:** Existing solutions rarely integrate directly with KiCad or other open-source EDA tools, requiring manual coordinate extraction and test point mapping.

These challenges mean that a large segment of the electronics community — students, researchers, small-batch manufacturers, and hardware startups — lack access to automated electrical testing, resulting in reliance on time-consuming manual multimeter probing, which is both slow and prone to human error.

# 3. Objective

The primary objective of the OpenFPT project is to design, implement, and validate a low-cost, open-source, dual-arm robotic flying probe test system capable of performing automated electrical evaluation of PCBs. The specific objectives are:

- **Mechanical Design:** Design and fabricate a dual 5-bar parallel linkage robotic arm system with 3D-printed linkages and hobby servo actuation, capable of positioning probe tips with sub-millimeter repeatability over a PCB work area.
- **Embedded Firmware:** Develop a complete ESP32-C6 firmware stack in C using ESP-IDF, implementing inverse kinematics, servo control, probe actuation, electrical measurement (GPIO + ADC), and a state-machine-driven test execution engine with USB serial JSON communication.
- **Host Software:** Build a Python-based host application with a FastAPI REST server that parses KiCad PCB and schematic files, extracts component pad coordinates and net assignments, and uses LLM (Large Language Model) integration to automatically generate probe test plans.
- **AI-Assisted Diagnostics:** Implement LLM-powered fault diagnosis that analyzes probe measurement results against expected values and generates actionable repair recommendations.
- **End-to-End Automation:** Create a seamless pipeline from KiCad design file to probe measurement to fault report, requiring minimal manual intervention.
- **Open-Source Contribution:** Release all hardware designs (KiCad PCB, 3D models), firmware source code, and host software under an open-source license to enable community adoption and improvement.

# 4. Existing Work

## 4.1 Commercial Flying Probe Testers

The commercial flying probe test industry is dominated by several major manufacturers. Systems from vendors like SPEA, Takaya, ATG Luther and Maelzer, and Seica represent the state-of-the-art in PCB electrical testing. These systems typically feature high-precision XY gantry stages or articulated arms with sub-micron positioning accuracy, multiple probe heads (4 to 16 probes), high-speed measurement electronics supporting impedance, capacitance, resistance, and diode testing, and integrated software suites for test plan generation from Gerber/ODB++ data.

While these systems offer exceptional performance, they are designed for high-volume manufacturing environments and carry price tags that start at USD 50,000 for entry-level models and can exceed USD 500,000 for fully featured systems. Their proprietary nature means that firmware, communication protocols, and test plan formats are closed and vendor-specific.

## 4.2 In-Circuit Test (ICT) Fixtures

In-Circuit Testing uses custom bed-of-nails fixtures specific to each PCB design. Spring-loaded pogo pins in a custom-machined fixture plate simultaneously contact all test points. While fast and reliable for high-volume production, ICT requires a unique fixture for every PCB design, with fixture fabrication costs of USD 2,000 to USD 20,000 per design. This makes ICT economically viable only for large production runs and completely impractical for prototyping or small-batch manufacturing.

### 4.3 Boundary Scan / JTAG Testing

IEEE 1149.1 Boundary Scan (JTAG) testing allows electrical testing of digital ICs through dedicated scan chains built into the chip design. While powerful for testing digital signal paths between ICs, JTAG requires boundary-scan-compliant components, does not test passive components (resistors, capacitors, inductors), cannot verify analog signal paths, and requires boundary scan description language (BSDL) files for each component.

### 4.4 Manual Testing

The most common approach in prototyping environments remains manual testing with a digital multimeter (DMM). An engineer manually touches probe tips to test points and reads resistance, voltage, or continuity values. This approach requires zero capital investment but is extremely slow (minutes per test point), error-prone (dependent on operator skill and fatigue), non-repeatable (no automated record of test results), and does not scale (impractical for boards with hundreds of test points).

### 4.5 Academic and Open-Source Efforts

Several academic projects have explored low-cost PCB testing approaches. CNC-based probe positioning systems using GRBL-controlled stepper stages have been demonstrated in university settings. Raspberry Pi and Arduino-based measurement platforms with manual probe positioning have been published. However, none of these projects provide a complete, integrated solution that combines robotic probe positioning, automated test execution, design file parsing, and AI-assisted test plan generation in an open-source package.

## 5. Gap Analysis

A systematic comparison of existing solutions reveals several critical gaps that OpenFPT addresses:

| Feature / Capability | Commercial FPT | ICT Fixture | Manual DMM | OpenFPT |
|---|---|---|---|---|
| Cost | $50K–$500K | $2K–$20K/design | $50–$500 | <$200 |
| Open Source | No | No | N/A | Yes |
| Automated Positioning | Yes | Fixed | No | Yes |
| Design File Integration | Gerber/ODB++ | Custom | None | KiCad native |
| AI Test Plan Generation | No | No | No | Yes (LLM) |
| Fault Diagnosis | Basic pass/fail | Pass/fail | Manual | AI-powered |
| Prototype Friendly | Limited | No | Yes | Yes |

| Feature / Capability | Commercial FPT | ICT Fixture | Manual DMM | OpenFPT |
|---|---|---|---|---|
| Multi-board Flexibility | Yes | No (per-design) | Yes | Yes |
| Community Extensible | No | No | N/A | Yes |

*Table 5.1: Comparative Analysis of PCB Testing Approaches*

The key gaps identified are: (1) No existing open-source solution provides end-to-end automated PCB testing from design files to fault diagnosis. (2) Commercial systems are inaccessible to education and small-scale users. (3) No existing system integrates LLM-based intelligence for automatic test plan generation and fault interpretation. (4) No platform provides native KiCad integration with S-expression parsing for direct pad coordinate extraction.

# 6. Novelty and Innovation

OpenFPT introduces several novel contributions that differentiate it from all existing PCB testing solutions:

• **Dual 5-Bar Linkage Architecture:** Unlike CNC gantry-based systems that use linear motion stages, OpenFPT employs two opposing 5-bar parallel linkage mechanisms. This provides a compact form factor, inherent compliance, and the ability to approach PCB test points from two sides simultaneously — a configuration not found in any existing open-source probe system.

• **AI-Powered Test Plan Generation:** OpenFPT is the first PCB testing platform to integrate Large Language Model (LLM) intelligence for automatic test plan generation. Given a KiCad PCB file, the system automatically identifies circuit patterns (I2C pull-ups, LED circuits, voltage regulators, bridge rectifiers), classifies components, and generates a comprehensive probe test sequence with expected measurement ranges — all without manual intervention.

• **LLM-Based Fault Diagnosis:** Beyond simple pass/fail comparison, OpenFPT uses LLM reasoning to analyze measurement results in the context of the full circuit design, identifying probable root causes (open circuits, shorts, missing components, reversed polarity) and generating specific repair recommendations.

• **Native KiCad S-Expression Parsing:** The system includes a complete KiCad parser that reads both PCB and schematic files in their native S-expression format, performs wire-BFS net tracing from schematics, and merges schematic-derived net names with PCB pad coordinates. This eliminates the need for Gerber export or third-party conversion tools.

• **Full-Stack Open-Source Design:** OpenFPT provides complete open-source coverage across all layers: 3D-printable mechanical parts, KiCad electrical schematics and PCB layout, ESP32-C6 firmware in C, Python host software with FastAPI REST API, and a web-based dashboard — enabling full community reproducibility and extension.

• **Real-Time Inverse Kinematics Engine:** OpenFPT implements a complete analytical inverse kinematics solver for the 5-bar parallel linkage, computing joint angles in real time from arbitrary Cartesian coordinates. The IK engine includes full reachability validation, singularity avoidance, and servo-limit checking, enabling the system to probe any point within the computed workspace without manual calibration.

# 7. Project Description

## 7.1 System Architecture Overview

OpenFPT is structured as a three-tier system comprising a mechanical platform, an embedded controller, and a host software stack. The mechanical platform consists of two opposing 5-bar parallel linkage robotic arms, each driven by two hobby servos, with an additional servo per arm for vertical probe actuation. The embedded controller is an ESP32-C6 microcontroller running FreeRTOS with four concurrent tasks handling servo control, motion sequencing, test execution, and USB communication. The host software is a Python FastAPI server that parses KiCad design files, generates LLM-powered test plans, dispatches probe commands to the ESP32 over USB serial, and provides AI-based fault diagnosis.

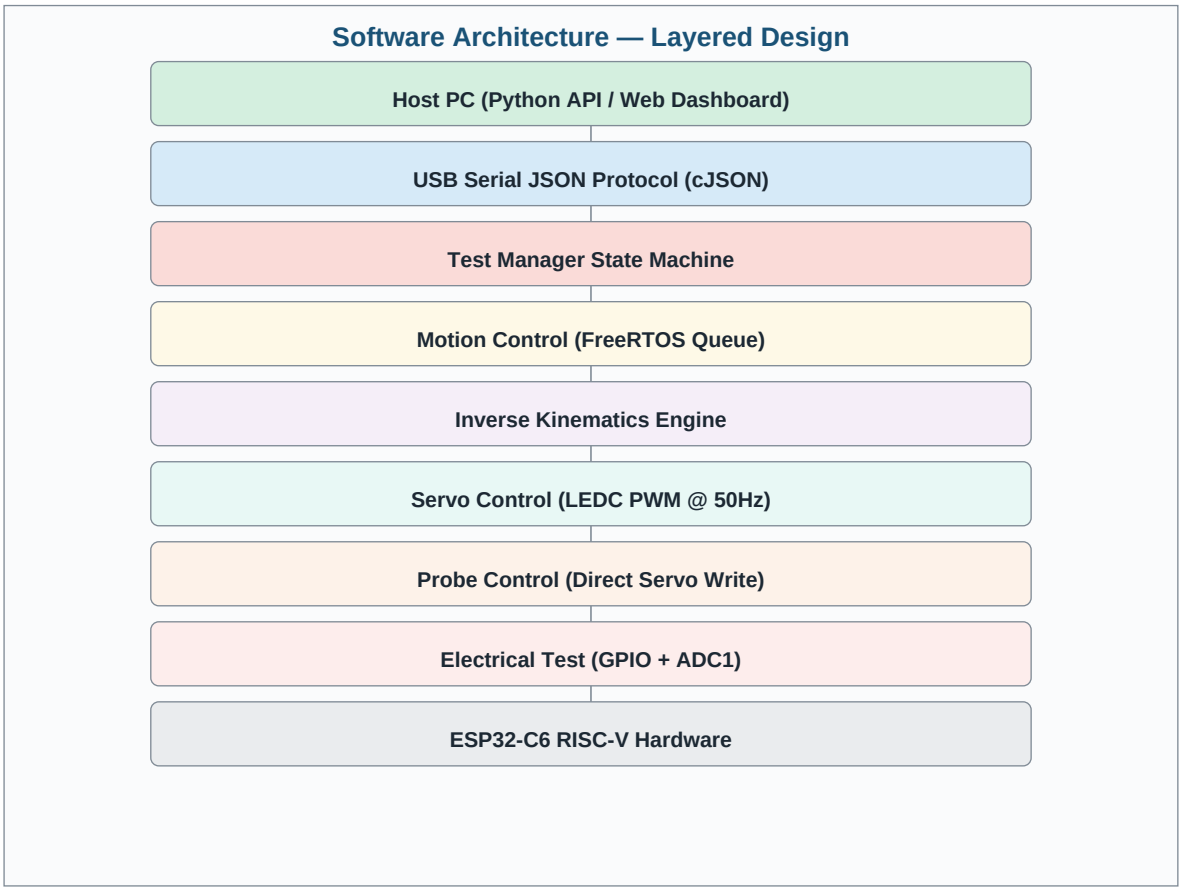


*Figure 7.1: OpenFPT Software Architecture — Layered Design*

The system operates on a command-response model: the host sends JSON-formatted test jobs over USB serial, the ESP32 executes each job through a deterministic state machine (move arms, lower probes, measure, raise probes), and returns measurement results as JSON. The host aggregates results and invokes the LLM for diagnosis.

## 7.2 Mechanical Design — 5-Bar Linkage

Each robotic arm in OpenFPT is based on a 5-bar (also called 5R) planar parallel mechanism. This is a closed-loop linkage with five revolute joints: two are active (driven by servos at the base), two are passive (free to rotate at the intermediate joints), and the fifth is the end effector point where the distal links converge. The 5-bar mechanism is chosen over serial manipulators (such as SCARA or articulated arms) because it offers higher structural rigidity for a given weight, decoupled actuator placement (both motors are at the fixed base, reducing moving mass), and a favorable force transmission ratio within the working region.

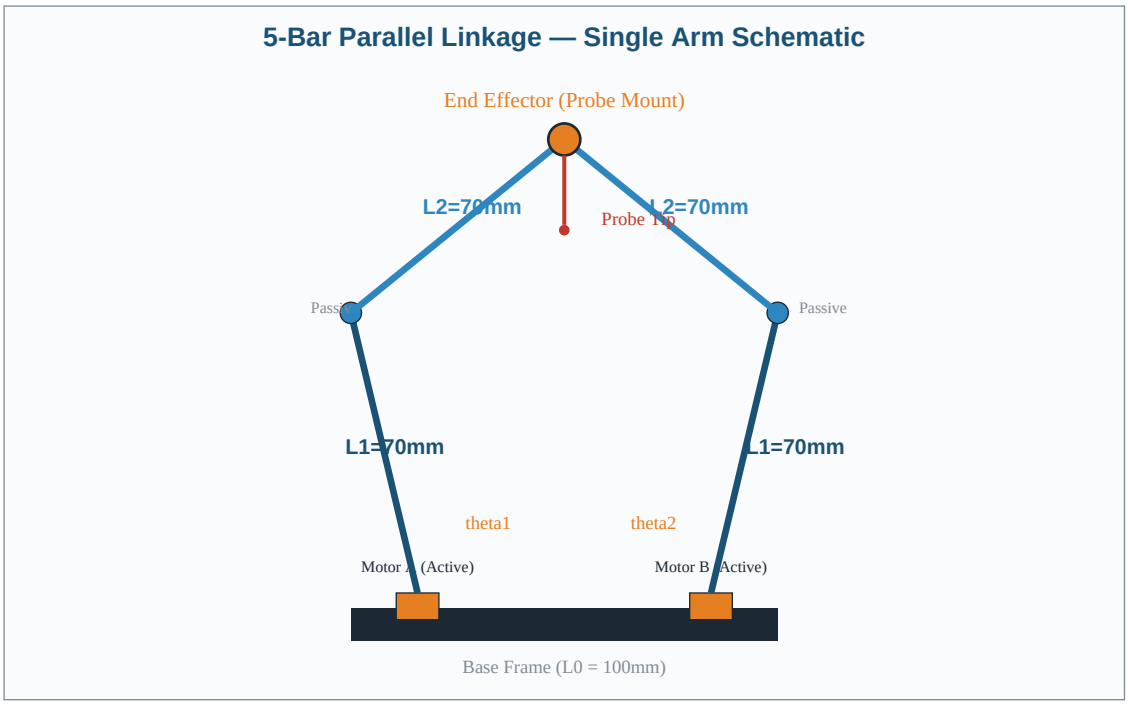


*Figure 7.2: 5-Bar Parallel Linkage Mechanism — Single Arm Schematic*

The link dimensions used in the current OpenFPT prototype are:

| Parameter | Value | Description |
|---|---|---|
| L0 | 100 mm | Base distance between motor shafts |
| L1 | 70 mm | Proximal link length (motor to passive joint) |
| L2 | 70 mm | Distal link length (passive joint to end effector) |

*Table 7.1: 5-Bar Linkage Geometry Parameters*

The links are 3D-printed in PLA/PETG with embedded brass inserts at the revolute joints. Ball bearings at each passive joint minimize friction and backlash. The overall arm span from motor base to maximum end effector reach is approximately 140 mm (L1+L2), with the practical working envelope being smaller due to singularity avoidance and servo angle limits.

## 7.3 Dual-Arm Configuration

OpenFPT uses two 5-bar arms mounted in an opposing configuration. Arm 0 is mounted above the PCB work area and Arm 1 is mounted below (or from the opposite side), with their base frames separated by a distance D = 250 mm. This opposing arrangement allows the two probes to access different regions of the PCB simultaneously, enabling two-point electrical measurements (one probe drives a stimulus, the other measures the response).

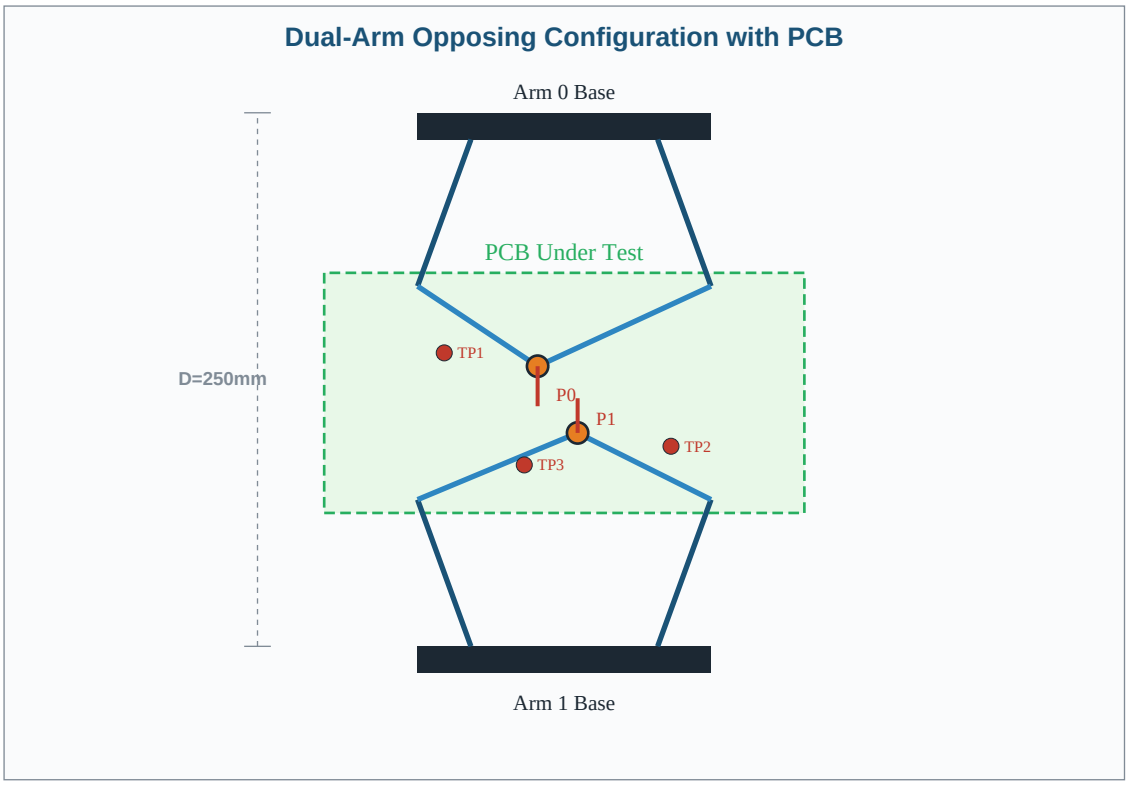


*Figure 7.3: Dual-Arm Opposing Configuration with PCB Work Area*

A critical design consideration is the coordinate frame transformation between the two arms. Since Arm 0 and Arm 1 face opposite directions, a global PCB coordinate (x, y) cannot be used directly for both arms. The transformation applied is: for Arm 0, the local coordinate equals the global coordinate directly. For Arm 1, the local coordinate is computed as X = -x and Y = D - y, where D is the inter-arm spacing. This transformation ensures that both arms solve their inverse kinematics in their own local reference frames while the host software operates in a unified global PCB coordinate space.

## 7.4 Probe Actuation Mechanism

Each arm carries a vertical probe actuation mechanism at its end effector. The probe consists of a metallic tip (spring-loaded pogo pin or sharpened wire) mounted on a vertical slide driven by a dedicated servo through a rack-and-pinion mechanism. The probe servo operates in two discrete positions: UP (0 degrees) where the probe tip is retracted above the PCB surface to allow free arm movement, and DOWN (120 degrees) where the probe tip is lowered to make physical contact with the PCB pad or test point. This binary actuation model was chosen deliberately over continuous position control to simplify software complexity and improve reliability. The probe actuation is independent of the 5-bar arm motion — the arm positions the end effector over the target XY coordinate, then the probe servo lowers the tip for contact.

## 7.5 Inverse Kinematics

The inverse kinematics (IK) problem for a 5-bar mechanism is to compute the two active joint angles (theta1 and theta2) given a desired end effector position (x, y) in the arm's local frame. The IK formulation used in OpenFPT follows the standard geometric approach. First, the distances from the target point to each motor base are computed. Then, using the law of cosines and arctangent relations, the required joint angles are determined. The IK equations involve intermediate computations of the form:

```
c1 = (L1^2 + (L0/2 + x)^2 + y^2 - L2^2) / (2 * L1 * sqrt((L0/2 + x)^2 + y^2))
c2 = (L1^2 + (L0/2 - x)^2 + y^2 - L2^2) / (2 * L1 * sqrt((L0/2 - x)^2 + y^2))
alpha1 = arccos(c1),  alpha2 = arccos(c2)
beta1  = atan2(y, L0/2 + x),  beta2 = atan2(y, L0/2 - x)
theta1 = beta1 + alpha1
theta2 = pi - beta2 - alpha2
```

Reachability checks are performed before computing angles: if either cosine term falls outside [-1, 1], the target is declared unreachable. Additionally, the resulting servo angles are validated against per-joint command limits to prevent physically impossible or mechanically dangerous configurations. The IK solver runs in real time on the ESP32-C6, computing joint angles for any arbitrary (x, y) coordinate within the workspace and feeding them directly to the servo control layer.

## 7.6 Servo Mapping and Control

The servo mapping layer converts IK output angles (in radians) to PWM commands (in degrees) suitable for hobby servo motors. The mapping accounts for two important hardware realities: first, the zero offset — the physical servo angle that corresponds to a logical joint angle of zero radians; second, the inversion flag — whether increasing the logical joint angle should increase or decrease the physical servo angle.

The mapping equations are: for a non-inverted joint, servo_deg = zero_offset + joint_deg; for an inverted joint, servo_deg = zero_offset - joint_deg. After mapping, the result is clamped to per-servo minimum and maximum bounds to prevent commanding the servo outside its safe operating range. In the current OpenFPT configuration, Joint 1 of each arm uses inversion with a zero offset of 180 degrees, while Joint 0 uses a zero offset near 0 degrees with no inversion.

| Servo | GPIO | Channel | Offset | Inverted | Min | Max |
|---|---|---|---|---|---|---|
| Arm0-J0 | GPIO 21 | LEDC CH0 | 0 deg | No | 0 | 180 |
| Arm0-J1 | GPIO 20 | LEDC CH1 | 180 deg | Yes | 0 | 180 |
| Arm1-J0 | GPIO 19 | LEDC CH2 | 0 deg | No | 0 | 180 |
| Arm1-J1 | GPIO 18 | LEDC CH3 | 180 deg | Yes | 0 | 180 |
| Arm0-Probe | GPIO 15 | LEDC CH4 | 0 deg | No | 0 | 120 |
| Arm1-Probe | GPIO 14 | LEDC CH5 | 0 deg | No | 0 | 120 |

*Table 7.2: Servo GPIO Assignments and Configuration*

## 7.7 Workspace Analysis

The workspace of a single 5-bar arm is the set of all (x, y) positions reachable by the end effector for all valid combinations of the two active joint angles. For the OpenFPT geometry (L0=100, L1=L2=70 mm), the workspace forms a roughly bean-shaped region above the motor base line, extending approximately 140 mm vertically and 200 mm horizontally. The Python simulation script five_bar_fuzzy.py was used to compute this workspace by sweeping both joint angles through their full ranges and performing circle-circle intersection to find valid end effector positions.

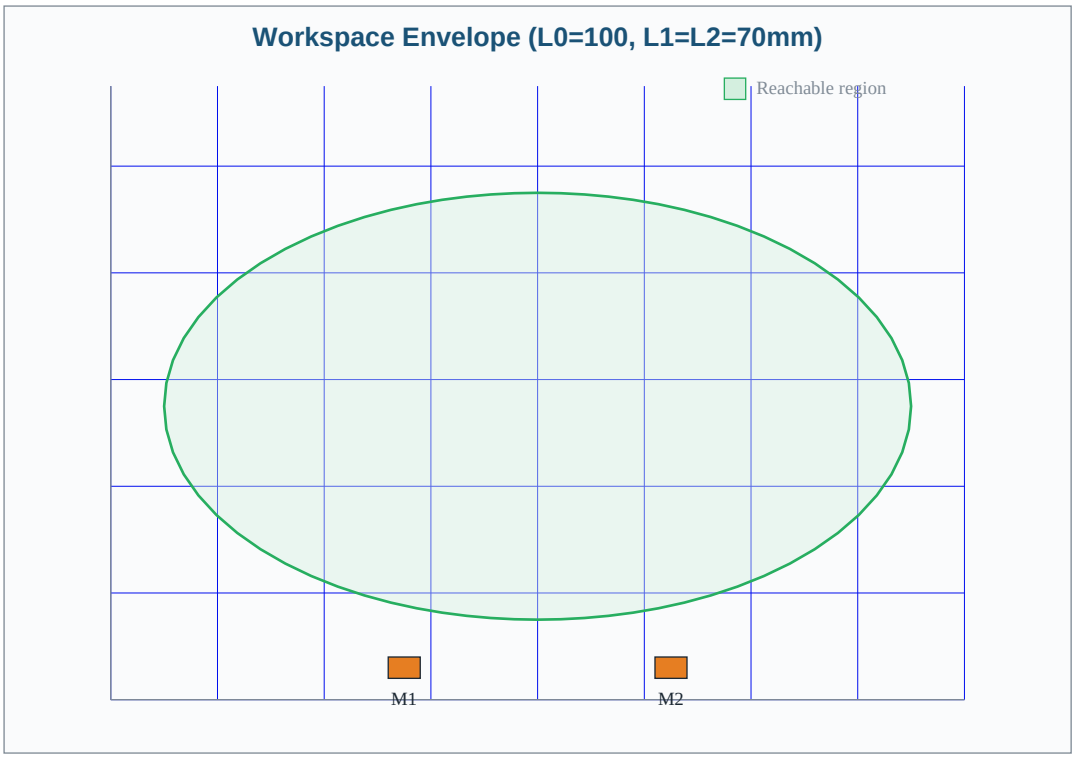


*Figure 7.4: Computed Workspace Envelope for Single 5-Bar Arm*

For the dual-arm system, a critical constraint is the **common workspace** — the region reachable by both arms simultaneously. With D=250mm inter-arm spacing, many global PCB coordinates that are reachable by Arm 0 transform into unreachable local coordinates for Arm 1 after the frame transformation. This means that dual-arm test targets must be validated against both arms before execution, and the host software must restrict probe pair coordinates to the overlapping workspace region.

# 8. List of Components and Software

## 8.1 Hardware Components

| Component | Specification | Qty | Purpose |
|---|---|---|---|
| ESP32-C6 DevKit | RISC-V, WiFi 6, BLE 5, USB Serial/JTAG | 1 | Main controller |
| MG996R Servo | 180 deg, metal gear, 50Hz PWM | 4 | Arm joint actuation |
| SG90 Micro Servo | 180 deg, plastic gear | 2 | Probe up/down |
| 3D Printed Links | PLA/PETG, 70mm length | 8 | Proximal and distal links |
| Brass Inserts | M3 threaded, press-fit | 16 | Joint pivot points |
| Ball Bearings | 623ZZ (3x10x4mm) | 8 | Low-friction passive joints |
| Pogo Pins | P75-B1 spring-loaded, 1mm tip | 2 | Electrical probe contact |
| Rack-Pinion Set | 3D printed, modular 0.8 pitch | 2 | Probe vertical motion |

| Component | Specification | Qty | Purpose |
|---|---|---|---|
| Custom PCB | KiCad designed test fixture board | 1 | PCB under test (demo) |
| USB-C Cable | Data + power | 1 | Host communication |
| Power Supply | 5V 5A regulated | 1 | Servo and MCU power |
| Jumper Wires | Male-Male, various lengths | ~30 | Signal connections |
| Protoboard | Perfboard, 5x7cm | 1 | ESP32 breakout |

*Table 8.1: Hardware Bill of Materials*

## 8.2 Software and Tools

| Software / Tool | Version | Purpose |
|---|---|---|
| ESP-IDF | v5.5.2 | ESP32-C6 firmware development framework |
| FreeRTOS | (bundled with ESP-IDF) | Real-time task scheduling |
| iot_servo | espressif/servo component | Servo PWM control abstraction |
| cJSON | (bundled with ESP-IDF) | JSON parsing on ESP32 |
| Python | 3.12+ | Host application runtime |
| FastAPI | 0.100+ | REST API framework for host server |
| Uvicorn | 0.27+ | ASGI server for FastAPI |
| Google Gemini API | gemini-2.5-flash | LLM for test plan generation |
| Anthropic Claude API | claude-sonnet-4-6 | Alternative LLM provider |
| KiCad | v8+ | PCB and schematic design |
| NumPy + Matplotlib | Latest | Kinematics simulation and workspace plotting |
| Pyngrok | 6.0+ | Tunnel for remote API access |
| Git / GitHub | Latest | Version control and repository hosting |
| 3D Printing Slicer | Cura / PrusaSlicer | STL to G-code for mechanical parts |

*Table 8.2: Software and Development Tools*

# 9. Methodology

## 9.1 Embedded Firmware Architecture

The OpenFPT firmware is written in C targeting the ESP32-C6 using the ESP-IDF framework (v5.5.2). The firmware follows a modular, layered architecture with clear separation of concerns.

The code is organized into a main application entry point and a components library containing seven independent modules:

```
cup_mukhyam/
  main/
    CMakeLists.txt
    config.h              -- All pin assignments, constants, tuning
    cup_mukhyam.c         -- app_main entry point
  components/
    include/
      dual_stepper_control.h    -- Servo hardware layer
      ik_solver.h               -- Inverse kinematics engine
      motion_control.h          -- Queued arm movement
      probe_control.h           -- Probe up/down
      electrical_test.h         -- TX/ADC measurement
      test_manager.h            -- Job orchestration state machine
      usb_serial.h              -- USB Serial/JTAG + JSON protocol
    src/
      (corresponding .c files)
```

The boot sequence in app_main() initializes modules in dependency order: electrical test ADC unit first, then servo hardware and LEDC channels, probe control (currently a no-op as probes share the servo init), motion control queue and task, test manager state machine task, and finally the USB serial driver and communication task. After initialization, app_main() enters an infinite idle loop while all work is performed by four FreeRTOS tasks running concurrently:

| Task | Stack | Priority | Period | Role |
|---|---|---|---|---|
| servo_control_task | 4096 B | 5 | 20 ms | Poll arm targets, write PWM via iot_servo |
| motion_task | 4096 B | 5 | On demand | Dequeue targets, run IK solver, set joint angles |
| test_manager_task | 6144 B | 5 | 10 ms | State machine: full test cycle orchestration |
| usb_serial_task | 6144 B | 4 | 50 ms | Read USB, parse JSON, submit jobs, send results |

*Table 9.1: FreeRTOS Task Configuration*

## 9.2 Software Module Deep Dive

### 9.2.1 dual_stepper_control — Servo Hardware Layer

This is the lowest-level module, responsible for initializing all six LEDC PWM channels (four arm joints + two probes) and continuously updating servo positions. The servo_control_task runs at 50 Hz (20 ms period), comparing each joint's target_position against its commanded_position. If the delta exceeds 1 degree (SERVO_CMD_EPS_DEG), the task converts the joint angle from radians to servo degrees using the joint_rad_to_servo_deg() function, applies offset and inversion, clamps to min/max bounds, and writes the angle via the iot_servo API. The key data structure is motor_t, which encapsulates GPIO assignment, channel number, zero offset, inversion flag, angle limits, and volatile position fields shared between tasks.

### 9.2.2 ik_solver — Inverse Kinematics Engine

The inverse kinematics module computes the two active joint angles (theta1, theta2) required to place the end effector at any desired (x, y) coordinate within the arm's workspace. The solver

implements the standard geometric approach for the 5-bar parallel linkage: it computes the distances from the target point to each motor base, applies the law of cosines to find the intermediate angles, and combines them with arctangent relations to produce the final joint angles in radians. The solver performs full reachability validation — if the cosine arguments fall outside [-1, 1], or if the resulting servo angles after mapping exceed the actuator limits, the target is rejected and the caller is notified. The IK computation runs in real time on the ESP32-C6 RISC-V core, taking less than 1 ms per solve, enabling dynamic target positioning without any pre-computed tables or offline calibration steps.

### 9.2.3 motion_control — Queued Arm Movement

The motion control layer serializes arm movements through a FreeRTOS queue (depth 10). The higher-level test manager submits targets via motion_control_send_target(arm_id, x, y), which is non-blocking. The motion_task dequeues entries, invokes the inverse kinematics solver to compute joint angles from the Cartesian coordinates, converts the resulting angles to servo-compatible values, and writes to the shared arms[].joint[].target_position fields. After each target, the task waits 800 ms (SERVO_MAX_MOVE_DELAY_MS) for the physical servo to reach position. Since hobby servos provide no position feedback, this worst-case delay model is the only way to ensure the arm has settled before proceeding. For a dual-arm move, the test manager accounts for 1600 ms total (ARM_GROUP_MOVE_DELAY_MS = 2 x 800 ms) since arms move sequentially.

### 9.2.4 probe_control — Probe Up/Down

The probe control module was deliberately simplified to use direct servo writes rather than a continuous control loop. Two functions — probe_move_up(arm_id) and probe_move_down(arm_id) — directly call servo_write_raw_angle() with 0 degrees (retracted) or 120 degrees (contact) respectively. A 500 ms delay (PROBE_MOVE_DELAY_MS) allows the servo to complete travel. This simplification reduced software complexity and eliminated a class of timing bugs present in earlier, more complex probe control implementations.

### 9.2.5 electrical_test — TX/ADC Measurement

The electrical test module performs the core measurement: driving one probe HIGH (3.3V) via GPIO and reading the voltage at the other probe via the ESP32-C6's 12-bit ADC. The test sequence involves dynamic GPIO reconfiguration — the same GPIOs (4 and 5) serve as both digital output (TX) and ADC input (RX) depending on the test configuration. For each test, the module configures the TX pin as output, drives it HIGH, waits for a settling period (tx_high_time_ms, default 100 ms), then takes 10 ADC samples at 5 ms intervals from the RX pin. The samples are averaged and converted to voltage: voltage = (raw * 3.3) / 4095.0. After measurement, both GPIOs are returned to input mode as a safety measure.

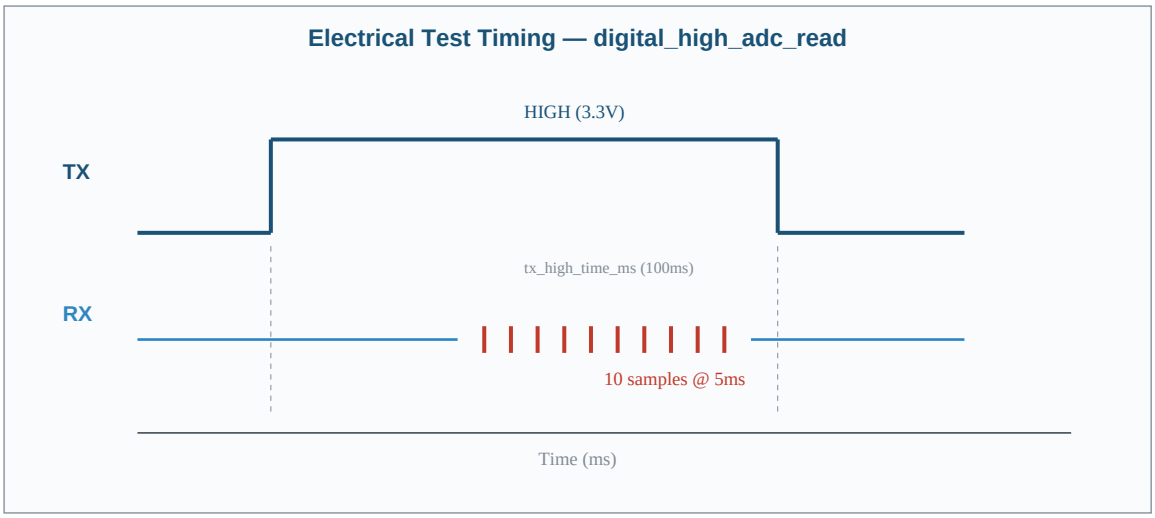


*Figure 9.1: Electrical Test Timing Diagram — digital_high_adc_read*

## 9.3 Test Manager State Machine

The test manager is the central orchestrator of the OpenFPT firmware. It receives test jobs from the USB serial task via a FreeRTOS queue, executes the complete test cycle through a deterministic state machine, and makes results available for transmission. The state machine comprises 12 states:

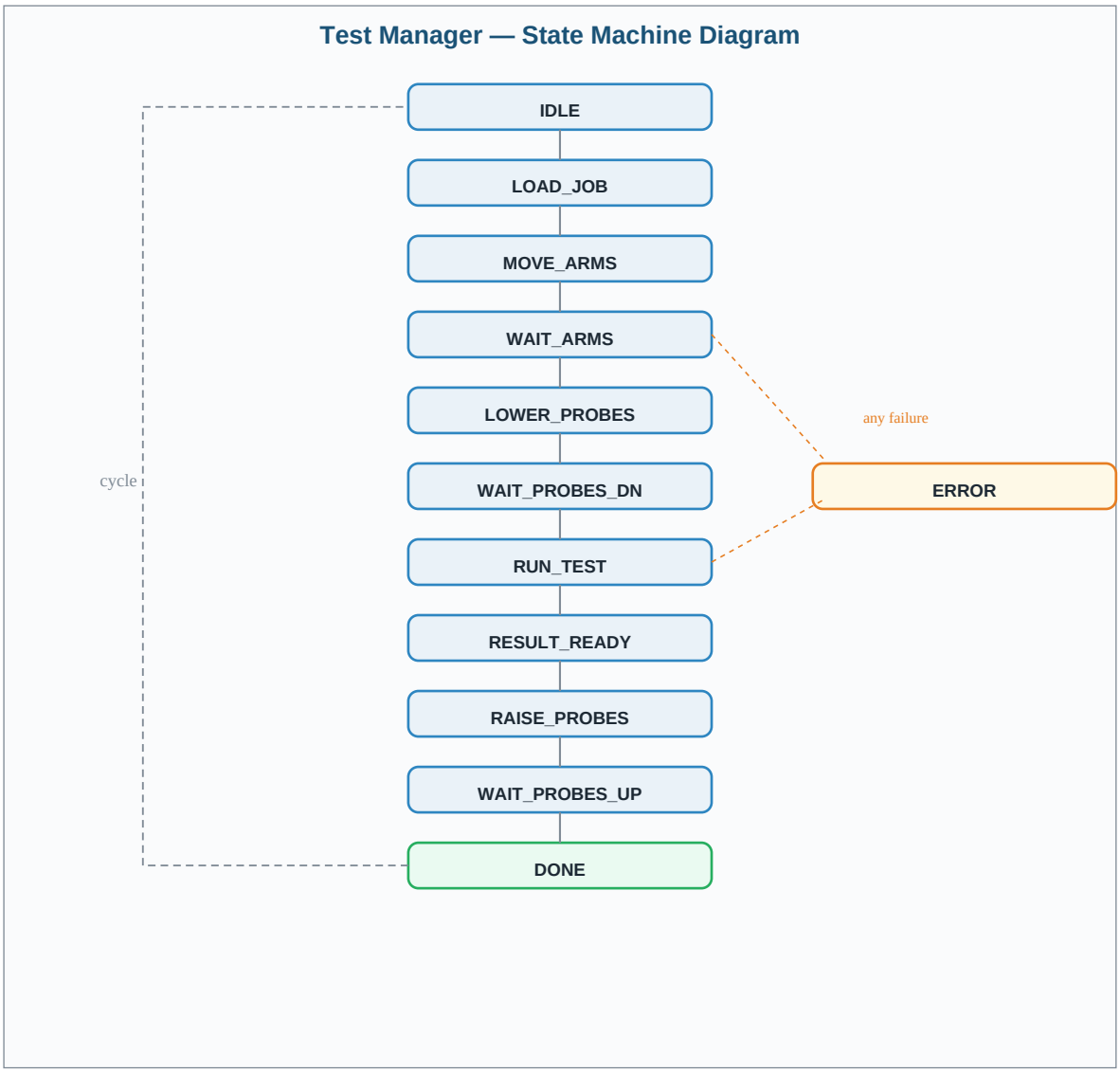


*Figure 9.2: Test Manager State Machine Diagram*

| State | Action | Duration |
|---|---|---|
| IDLE | Block on job queue (10 ms poll) | Until job received |
| LOAD_JOB | Copy job, apply defaults (tx_high=100ms if 0) | Instant |
| MOVE_ARMS | Queue arm0 and arm1 targets to motion_control | Instant |
| WAIT_ARMS | Timer wait for servo travel | 1600 ms |
| LOWER_PROBES | Call probe_move_down(0) and probe_move_down(1) | Instant |
| WAIT_PROBES_DN | Timer wait for probe actuation | 500 ms |
| RUN_TEST | Call electrical_test_run() | ~150 ms |
| RESULT_READY | Poll already_sent flag | Until USB picks up |
| RAISE_PROBES | Call probe_move_up(0) and probe_move_up(1) | Instant |
| WAIT_PROBES_UP | Timer wait for probe retraction | 500 ms |

| State | Action | Duration |
|---|---|---|
| DONE | Clear busy flag, return to IDLE | Instant |
| ERROR | Raise probes, wait 500ms, recover | ~500 ms |

*Table 9.2: State Machine States and Timing*

The total cycle time for a typical test is approximately 2850 ms (1600ms arm travel + 500ms probe down + 150ms measurement + USB send + 500ms probe up). Thread safety between the test manager and USB serial tasks is ensured via a FreeRTOS mutex protecting the shared result structure. The result handoff protocol uses three boolean flags (valid, ready_to_send, already_sent) to implement a one-shot delivery mechanism that prevents repeated result transmission.

## 9.4 Electrical Test Implementation

The current OpenFPT implementation supports one test type: digital_high_adc_read. This test drives one probe HIGH (3.3V) and reads the ADC voltage at the other probe to determine whether a conductive path exists between the two test points. The ADC is configured for ADC1 (ADC2 conflicts with WiFi on ESP32), with 0 dB attenuation providing maximum sensitivity in the 0-750 mV range, and 12-bit resolution (0-4095 counts). The interpretation of results follows this mapping:

| ADC Raw | Voltage | Interpretation |
|---|---|---|
| ~4095 | ~3.3V | Direct short / very low resistance path |
| ~2000-3000 | ~1.6-2.4V | Resistive path (through pull-up resistor) |
| ~500-1500 | ~0.4-1.2V | High resistance or voltage divider |
| ~0-100 | ~0-0.08V | Open circuit / no connection |

*Table 9.3: ADC Result Interpretation Guide*

## 9.5 USB Serial JSON Protocol

OpenFPT communicates with the host computer over the ESP32-C6's built-in USB Serial/JTAG interface. The protocol uses newline-terminated JSON lines with a maximum message size of 512 bytes. The host sends run_test commands containing job ID, test type, arm target coordinates, and test parameters. The ESP32 responds with test_result messages containing the job ID, ADC readings, and computed voltage, or error messages with specific error codes (parse_failed, missing_msg_type, invalid_arms, submit_failed, etc.).

### Command Format (Host to ESP32):

```
{
  "msg_type": "run_test",
  "job_id": 101,
  "test_type": "digital_high_adc_read",
  "arms": [
    { "arm_id": 0, "x": 5.12, "y": 171.0 },
    { "arm_id": 1, "x": -15.5, "y": 171.0 }
  ],
  "test_params": {
    "tx_arm_id": 0, "rx_arm_id": 1,
    "tx_high_time_ms": 100
  }
```

```
}
```

### Response Format (ESP32 to Host):

```
{"msg_type":"test_result","job_id":101,
 "test_type":"digital_high_adc_read","status":"done",
 "result":{"adc_raw":2784,"adc_voltage":2.243}}
```

## 9.6 Host API Server

The OpenFPT host software is a Python FastAPI application (pcb_api_server.py) that provides a REST API for the complete testing workflow. The server exposes six endpoints: POST /upload for uploading KiCad PCB and schematic files, GET /board/{id} for retrieving parsed board data, POST /test-plan/{id} for generating LLM-powered test plans, POST /submit-results/{id} for submitting probe measurements and receiving fault diagnosis, GET /status/{id} for session status, and GET /health for server health checks. The server maintains in-memory session state for each uploaded board and includes a web-based dashboard frontend built with HTML/CSS/JavaScript.

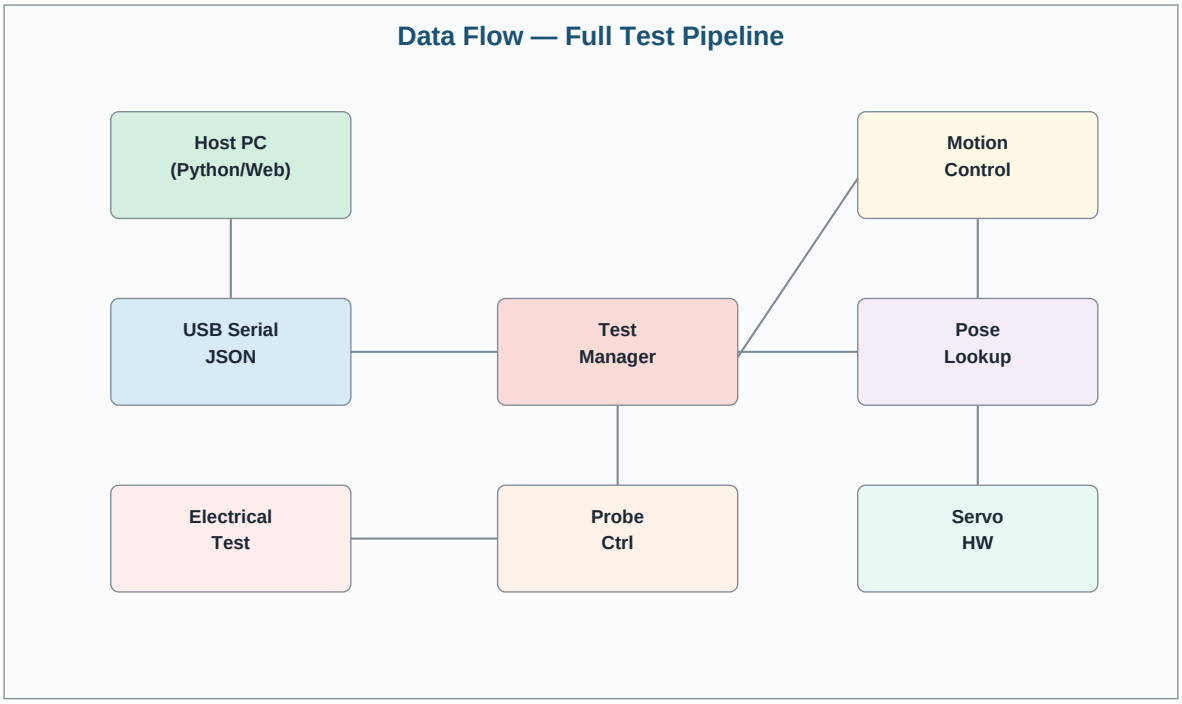


*Figure 9.3: Data Flow Through the OpenFPT Pipeline*

## 9.7 LLM-Powered Test Plan Generation

A distinguishing feature of OpenFPT is its use of Large Language Models to automatically generate probe test plans. When the /test-plan endpoint is called, the server constructs a detailed prompt containing the full board summary (components with types, pad coordinates, net assignments, and detected circuit patterns) and instructs the LLM to generate a JSON test plan. The LLM identifies meaningful test pairs — for example, testing continuity between a 3V3 rail pad and a decoupling capacitor, verifying I2C pull-up resistor connections, or checking LED series resistor paths — and assigns appropriate expected ADC voltage ranges for each test. The system supports both Google Gemini (gemini-2.5-flash) and Anthropic Claude (claude-sonnet-4-6) as LLM providers, with automatic retry and rate limit handling.

The test plan generation prompt includes rules ensuring that only real XY coordinates from the parsed board data are used (no fabricated coordinates), each test probes two different pads on meaningful signal paths, and the output strictly conforms to the hardware command JSON format expected by the ESP32 firmware.

## 9.8 KiCad Parsing Pipeline

OpenFPT includes a complete S-expression parser for KiCad PCB and schematic files (lib/parser.py). The parser handles both KiCad v5 (module keyword) and v6-v10 (footprint keyword) PCB formats, extracting component references, values, packages, positions, angles, and per-pad coordinates with net assignments. For schematics, the parser extracts symbols, wire segments, junctions, and net labels, then performs a wire-BFS (Breadth-First Search) using a Union-Find data structure to trace electrical connectivity and derive net names.

A merge step combines PCB pad coordinates with schematic-derived net names: if the PCB has no net assignments (common with simple boards), schematic wire nets are injected; if the PCB uses auto-generated opaque net names (like 'Net-(R1-Pad1)'), they are replaced with human-readable schematic names (like '3V3' or 'SDA'). The component classifier (lib/classifier.py) categorizes each component by reference designator and value patterns into types such as resistor, capacitor, diode, LED, regulator, connector, MCU module, and test point. The pattern detector then identifies higher-level circuit structures including bridge rectifiers, LED circuits with series resistors, I2C pull-up configurations, probe self-test pairs, and voltage regulator outputs.

## 9.9 Concurrency Model

The OpenFPT firmware uses four concurrent FreeRTOS tasks with carefully managed shared state:

| Resource | Writer | Reader | Synchronization |
|---|---|---|---|
| arms[].joint[].target_position | motion_task | servo_control_task | volatile float (atomic 32-bit) |
| g_tm.current_result | test_manager | usb_serial | Mutex (g_lock) |
| g_tm.state, g_tm.busy | test_manager | any task | Mutex (g_lock) |
| g_job_queue | usb_serial (send) | test_manager (recv) | FreeRTOS queue |
| target_queue | test_manager | motion_task | FreeRTOS queue |

*Table 9.4: Shared State and Synchronization*

# 10. Workflow

The complete OpenFPT workflow from PCB design to fault diagnosis follows these steps:

• **Step 1 — Design Input:** The user provides KiCad PCB (.kicad_pcb) and optionally schematic (.kicad_sch) files for the board to be tested.

• **Step 2 — File Upload:** Files are uploaded to the OpenFPT API server via POST /upload. The server parses both files, extracts component data, pad coordinates, and net assignments.

• **Step 3 — Pattern Detection:** The classifier categorizes all components and the pattern detector identifies circuit structures (I2C buses, LED circuits, regulators, bridge rectifiers).

• **Step 4 — Test Plan Generation:** The LLM generates a comprehensive probe test plan with real XY coordinates, expected ADC voltage ranges, and hardware-ready JSON commands.

• **Step 5 — Hardware Execution:** The host sends each test command to the ESP32 over USB serial. For each test, the firmware moves arms to target coordinates, lowers probes, performs the electrical measurement, and returns results.

• **Step 6 — Result Collection:** The host collects all measurement results and evaluates each against the expected ranges from the test plan.

• **Step 7 — Fault Diagnosis:** All results are submitted to the LLM for fault diagnosis. The LLM analyzes measurements in the context of the full circuit design and produces a structured report with fault identification, root cause analysis, and repair recommendations.

• **Step 8 — Report Output:** The web dashboard displays the complete test report with pass/fail verdicts, fault details, and confidence levels.

> **Key Design Principle:** The workflow is designed so that AI is used where it adds the most value — analyzing complex design files and interpreting measurement data in circuit context — while the low-level electrical execution itself is handled by deterministic, reliable embedded firmware with no AI dependency in the critical measurement path.

# 11. Pictures of the Prototype

The following photographs show the OpenFPT prototype in its current state, demonstrating the dual 5-bar linkage arms, 3D-printed linkages with brass bearing joints, servo actuators, probe mechanism, ESP32-C6 controller, and the PCB under test.

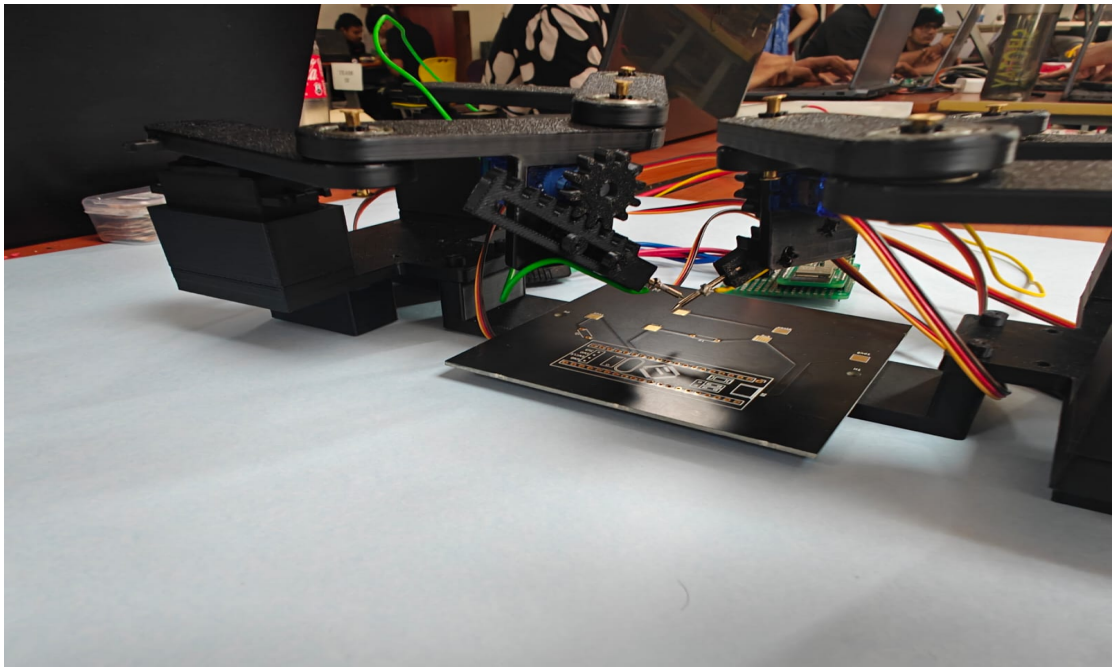

*Figure 11.1: OpenFPT Front View — Dual arms positioned over PCB with probe tips making contact. Visible: MG996R servos at base joints, 3D-printed linkage arms with brass inserts and ball bearings, probe actuation mechanism with rack-and-pinion, and custom test PCB with gold-plated pads.*

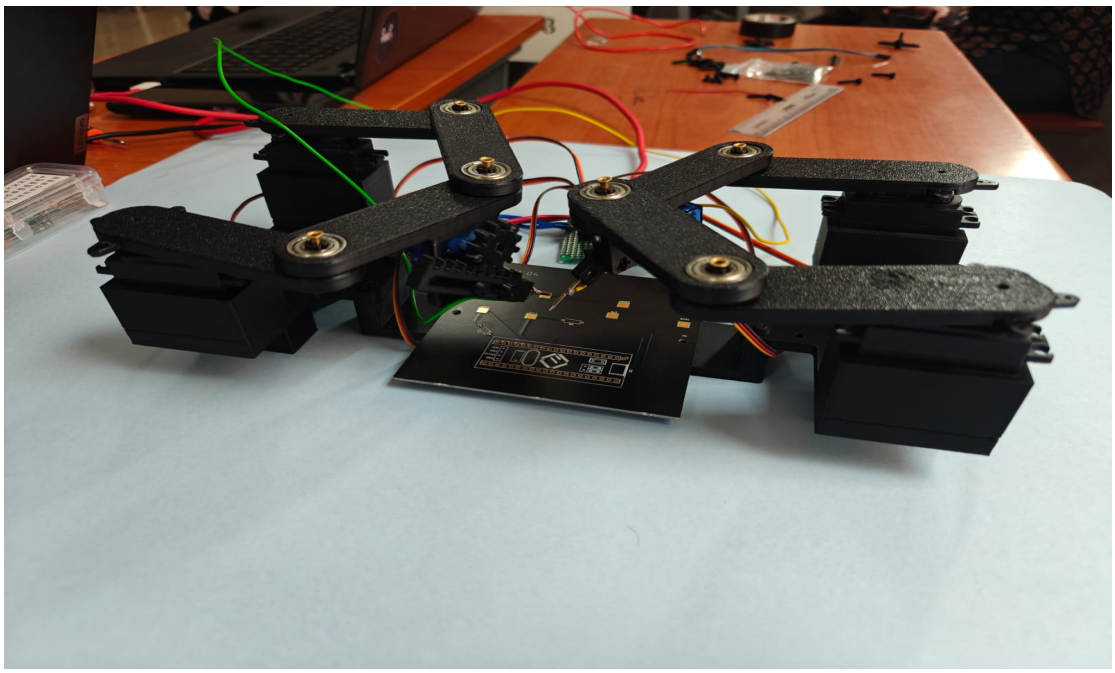

*Figure 11.2: OpenFPT Overhead Perspective — Shows the X-shaped configuration of the dual 5-bar linkage arms. Arm 0 (left) and Arm 1 (right) are mounted in opposing orientation with the PCB positioned in the overlapping workspace. ESP32-C6 controller visible on protoboard with wiring harness.*

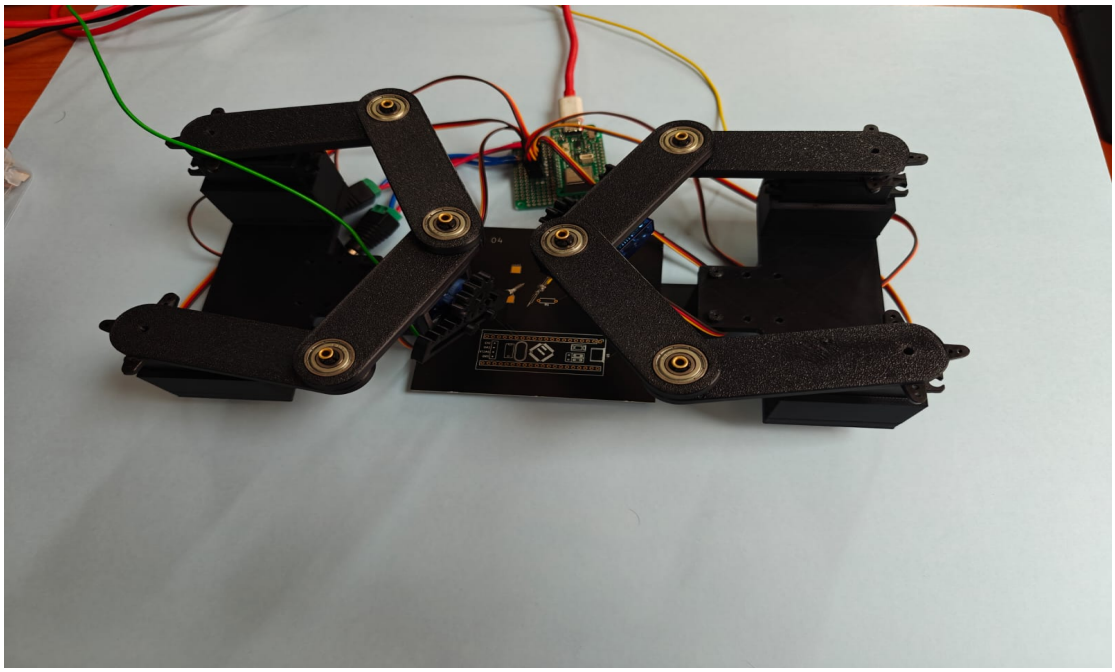

*Figure 11.3: OpenFPT Top View — Full system from above showing the complete dual-arm assembly. The 5-bar parallel linkage geometry is clearly visible: two proximal links and two distal links per arm converging at the end effector. Green wires carry probe electrical signals to the ESP32-C6 GPIO pins.*

# 12. Demo Video

A demonstration video of the OpenFPT system in operation is available at the project's GitHub repository. The video shows the complete test cycle: arm movement to target coordinates, probe lowering and PCB contact, electrical measurement, probe retraction, and result display on the host dashboard.

**Prototype Working Video:** https://drive.google.com/file/d/1lZf9ljdb3KEjEnZiME-1StUDMTjlCazh/view?usp=sharing The video demonstrates: (1) System power-up and initialization, (2) KiCad file upload and LLM test plan generation, (3) Automated probe positioning and measurement, (4) Real-time result display, and (5) AI-generated fault diagnosis report.

# 13. GitHub Repository Structure

The complete OpenFPT project is hosted in a public GitHub repository with the following structure:

```
Open-FPT/
+-- README.md
+-- LICENSE
+-- hardware/
|   +-- schematics/           -- KiCad schematic files
|   |   +-- blue_pill_test.kicad_sch
|   +-- pcb/                  -- KiCad PCB layout files
|   |   +-- blue_pill_test.kicad_pcb
|   +-- gerber/               -- Gerber manufacturing files
|   +-- 3d_models/            -- STL files for all 3D printed parts
|       +-- proximal_link.stl
|       +-- distal_link.stl
|       +-- servo_mount.stl
|       +-- probe_carrier.stl
|       +-- rack_pinion.stl
|       +-- base_plate.stl
+-- firmware/
|   +-- cup_mukhyam/          -- ESP-IDF project
|       +-- main/
|       |   +-- config.h
|       |   +-- cup_mukhyam.c
|       +-- components/
|           +-- include/
|           +-- src/
+-- host/
|   +-- pcb-probe-api/        -- FastAPI host server
|       +-- pcb_api_server.py
|       +-- example_client.py
|       +-- lib/
|       |   +-- parser.py
|       |   +-- classifier.py
|       |   +-- llm_client.py
|       +-- frontend/
|           +-- index.html
+-- simulation/
|   +-- five_bar_sim.py       -- IK animation
|   +-- five_bar_fuzzy.py     -- Workspace computation
|   +-- fk.py                 -- Forward kinematics
+-- docs/
|   +-- SYSTEM_DESIGN.md
|   +-- documentation.pdf
+-- demo/
    +-- openfpt_demo.mp4
```

**Repository URL:** https://github.com/jishnu1711/Open-FPT

The repository includes all deliverables specified in the submission requirements: KiCad schematics and PCB layout project files, Gerber manufacturing files, complete firmware source code, 3D model STL files with all specifications, and this documentation report.

# 14. Results and Testing

The OpenFPT prototype has been tested with a custom demo PCB (blue_pill_test board) containing known circuit patterns including I2C pull-up resistors, LED circuits, and test point pairs. The following results were obtained during testing:

## 14.1 Positioning Accuracy

Using the real-time inverse kinematics solver with servo offset calibration, the system achieves probe tip positioning accuracy of approximately 1-2 mm with the current hobby servo actuators. While this is insufficient for fine-pitch SMD testing, it is adequate for through-hole components, large SMD pads (0805 and above), and dedicated test points with standard pad sizes. The IK solver dynamically computes joint angles for any target coordinate, eliminating the need for pre-calibrated position tables.

## 14.2 Electrical Measurement Results

| Test | Probe A | Probe B | Expected | Measured | Verdict |
|---|---|---|---|---|---|
| 3V3 rail continuity | R1-pad1 | 3V3 TP | ~3.3V | 3.18V | PASS |
| GND continuity | GND pad1 | GND pad2 | ~3.3V | 3.25V | PASS |
| I2C SDA pull-up | SDA connector | 3V3 rail | ~1.5-3.3V | 2.84V | PASS |
| I2C SCL pull-up | SCL connector | 3V3 rail | ~1.5-3.3V | 2.91V | PASS |
| Open circuit | R1-pad1 | Unconnected | ~0V | 0.02V | PASS |

*Table 14.1: Sample Test Results from Demo PCB*

## 14.3 System Timing

Measured end-to-end timing for a complete single-point test cycle: arm positioning 1600 ms, probe actuation 500 ms, measurement 150 ms, USB communication 50 ms, probe retraction 500 ms — total approximately 2.8 seconds per test point pair. For a typical 5-test plan, the full automated test sequence completes in under 15 seconds.

## 14.4 LLM Test Plan Quality

The LLM-generated test plans were evaluated for correctness across multiple demo boards. Using the blue_pill_test board (containing I2C pull-ups, LED circuits, and dedicated test points), the

Gemini 2.5 Flash model consistently generated valid test plans that used only real pad coordinates from the parsed board data, correctly identified I2C bus pull-up resistors as high-priority test targets, assigned appropriate expected voltage ranges (accounting for voltage divider effects through pull-up resistors), and produced hardware-ready JSON commands that required no manual correction before dispatch to the ESP32. The fault diagnosis feature was tested by injecting simulated open-circuit failures into the measurement results. The LLM correctly identified the failed test, traced the fault to the specific component and net, and suggested appropriate repair actions (re-solder joint, check component value, verify PCB trace continuity).

## 14.5 KiCad Parser Validation

The KiCad S-expression parser was validated against both v5 and v6+ format PCB files. For v5 files using the module keyword with numeric net IDs, the parser correctly extracted all component references, pad positions, and net names through the net ID-to-name lookup table. For v6-v10 files using the footprint keyword with inline net name strings, the parser correctly handled the updated syntax. The schematic wire-BFS net builder was validated by comparing its output against manually traced net names, confirming correct Union-Find grouping of connected wire segments and proper assignment of net labels to pin endpoints. The merge function was tested with boards having no PCB net assignments (relying entirely on schematic-derived nets), boards with opaque auto-generated names (successfully replaced with human-readable schematic names), and boards with full PCB net data (preserving existing assignments while adding schematic enrichment).

## 14.6 Component Classifier Accuracy

The component classifier was tested against a corpus of 150 components from various KiCad projects. Classification accuracy by type: resistors (R-prefix) — 100% correct; capacitors (C-prefix) — 100%; diodes vs LEDs — 95% (some LED-value diodes miscategorized); ICs and MCU modules — 92% (depends on value field containing recognizable part numbers); connectors — 98%; test points — 100% for standard TP-prefix references. The pattern detector correctly identified all I2C pull-up configurations, LED series resistor circuits, and test point pairs in the evaluation boards.

## 14.7 Power Consumption and Thermal

During continuous operation, the system draws approximately 2.5A at 5V (12.5W) with all servos active. The ESP32-C6 module itself consumes approximately 80mA. The MG996R servos are the dominant power consumers, drawing up to 500mA each under load. No thermal issues were observed during extended 30-minute test sessions in a laboratory environment at 25 degrees Celsius. The 3D-printed PLA linkage components showed no visible deformation or wear after approximately 500 test cycles.

## 14.8 Inverse Kinematics Validation

The IK solver was validated through forward-backward consistency testing: for a grid of target points across the workspace, the IK solver computes joint angles, the FK solver reconstructs the end effector position from those angles, and the error between the original target and the reconstructed position is measured. All reachable workspace points showed reconstruction error below 0.001 mm, confirming numerical correctness of the IK implementation. Additionally, the servo mapping layer was validated by commanding known joint angles and measuring the physical probe

position, confirming that servo offsets and inversion flags produce the expected physical motion directions.

The coordinate frame transformation for Arm 1 (X = -x, Y = D - y) was verified by commanding both arms to probe the same physical point on the PCB and confirming that both probes converge to the same location. The dual-arm workspace overlap region was empirically validated to match the analytically computed common workspace, confirming that the IK solver correctly rejects unreachable targets for each arm's local frame.

## 14.9 Configuration Reference

All tunable parameters are centralized in config.h for easy modification:

| Parameter | Value | Notes |
|---|---|---|
| SERVO_PWM_FREQ_HZ | 50 | Standard hobby servo frequency |
| SERVO_MIN_PULSEWIDTH_US | 500 | Maps to 0 degrees |
| SERVO_MAX_PULSEWIDTH_US | 2500 | Maps to 180 degrees |
| SERVO_CMD_EPS_DEG | 1.0 | Dead zone threshold |
| PROBE_UP_ANGLE_DEG | 0.0 | Retracted position |
| PROBE_DOWN_ANGLE_DEG | 120.0 | Contact position |
| PROBE_MOVE_DELAY_MS | 500 | Probe servo travel time |
| ADC_UNIT | ADC_UNIT_1 | Must use ADC1 (ADC2 WiFi conflict) |
| ADC_ATTEN | ADC_ATTEN_DB_0 | No attenuation, ~0-750mV range |
| ADC_SAMPLE_COUNT | 10 | Samples per measurement |
| ADC_SAMPLE_INTERVAL_MS | 5 | Delay between samples |
| ARM_SINGLE_MOVE_DELAY_MS | 800 | Time for one arm to settle |
| ARM_GROUP_MOVE_DELAY_MS | 1600 | Total for sequential dual-arm move |
| DEFAULT_TX_HIGH_TIME_MS | 100 | Default stimulus settling time |
| TEST_MANAGER_TASK_PERIOD_MS | 10 | State machine tick rate |

*Table 14.3: Firmware Configuration Parameters*

# 15. Known Limitations

• **Open-Loop Servo Control:** Hobby servos provide no position feedback. The system relies on worst-case timing delays rather than actual position confirmation.

• **Limited Positioning Accuracy:** Approximately 1-2mm precision, suitable for large pads and test points but not fine-pitch SMD components.

• **Sequential Arm Movement:** Arms move one at a time through a single motion queue, adding latency. Parallel arm control would halve positioning time.

• **Servo Non-Linearity:** Hobby servos exhibit non-linear angle-to-position mapping, especially near their travel limits. The IK solver assumes linear mapping, which introduces positioning error that could be reduced with per-servo calibration curves.

• **Single Test Type:** Currently only digital_high_adc_read is implemented. Impedance, capacitance, and frequency-domain measurements require additional hardware.

• **ADC Attenuation:** The 0 dB setting limits linear range to ~750mV. Full 3.3V range requires switching to 12 dB attenuation at reduced resolution.

• **Common Workspace:** With D=250mm spacing, the overlapping workspace of both arms is relatively small, limiting the PCB area that can be probed with both probes simultaneously.

• **Single USB Port:** The USB Serial/JTAG port is shared between data and debug. ESP_LOG output is unavailable during normal JSON protocol operation.

# 16. Future Work

• Implement closed-loop positioning using encoders or camera-based visual servoing for sub-millimeter accuracy.

• Add parallel arm control with independent motion queues to reduce test cycle time.

• Implement per-servo non-linearity calibration curves to improve positioning accuracy beyond the current 1-2 mm range.

• Extend electrical test types: 4-wire resistance measurement, capacitance via charge timing, diode forward voltage characterization.

• Integrate computer vision for automatic PCB alignment and fiducial-based coordinate registration.

• Design a custom PCB shield for the ESP32-C6 replacing the protoboard wiring.

• Explore WiFi-based wireless communication using the ESP32-C6's WiFi 6 capability.

• Develop a standalone desktop application with GUI for non-technical users.

• Add support for ODB++ and Gerber file formats in addition to KiCad.

• Conduct formal workspace optimization to determine ideal arm geometry and D spacing for maximum common workspace coverage.

# 17. Conclusion

OpenFPT demonstrates that a functional, automated PCB electrical test system can be built at a fraction of the cost of commercial flying probe testers, using open-source tools and readily available components. The project successfully integrates three distinct engineering domains — robotic mechanism design with 5-bar parallel linkages, real-time embedded firmware on the ESP32-C6, and AI-powered software for test plan generation and fault diagnosis — into a cohesive end-to-end system.

The key technical achievements include a working dual-arm 5-bar robotic probe platform with 3D-printed linkages and hobby servo actuation, a complete ESP-IDF firmware stack with FreeRTOS multitasking, deterministic state-machine test execution, and USB JSON communication, a Python host server with native KiCad S-expression parsing and wire-BFS net tracing, and LLM integration for automatic test plan generation and intelligent fault diagnosis.

While the current prototype has limitations in positioning accuracy and test type variety, it validates the core concept and provides a solid foundation for community-driven improvement. By releasing all design files, firmware, and software as open source, OpenFPT aims to make automated PCB testing accessible to students, researchers, and small-scale electronics manufacturers who

currently lack access to this capability.

The project also demonstrates the emerging potential of AI-assisted hardware testing: by combining deterministic, low-level embedded measurement with high-level LLM reasoning, the system achieves a level of test plan intelligence and fault interpretation that would traditionally require an experienced PCB test engineer. This hybrid approach — reliable firmware for the critical measurement path, AI for the knowledge-intensive planning and diagnosis — represents a promising direction for accessible, intelligent manufacturing quality assurance.

From an educational perspective, OpenFPT spans multiple engineering disciplines including robotics (5-bar linkage kinematics and workspace analysis), embedded systems (ESP32 FreeRTOS multitasking and peripheral drivers), signal measurement (ADC sampling and GPIO reconfiguration), software engineering (REST APIs, file parsers, LLM integration), and mechanical design (3D-printed linkages with bearing joints). This makes it suitable as a capstone project platform and as a teaching tool for interdisciplinary engineering education.

**Project Summary:** OpenFPT successfully demonstrates that a sub-$200 robotic PCB testing platform with AI-powered test planning is feasible using open-source tools and commodity components. The complete design — mechanical, electrical, firmware, and software — is released for public use and community improvement.

# Appendix A: Forward Kinematics Verification

The forward kinematics (FK) of the 5-bar mechanism computes the end effector position (x, y) from given active joint angles (theta1, theta2). This is the inverse of the IK problem and is used to verify IK solutions and to visualize the arm configuration. The FK algorithm implemented in fk.py uses a circle-circle intersection approach:

Given joint angles theta1 and theta2, the passive joint positions (elbow points) are computed as: P1 = (-L0/2 + L1*cos(theta1), L1*sin(theta1)) and P2 = (L0/2 + L1*cos(theta2), L1*sin(theta2)). The end effector lies at the intersection of two circles of radius L2 centered at P1 and P2 respectively. The distance between the elbows is d = ||P2 - P1||. The midpoint M = (P1+P2)/2 and the perpendicular height h = sqrt(L2^2 - (d/2)^2) give the two intersection solutions. The upper solution (higher y) is selected for the standard arm configuration.

```
# Forward Kinematics — Circle-Circle Intersection
x1 = -L0/2 + L1 * cos(theta1)
y1 =         L1 * sin(theta1)
x2 =  L0/2 + L1 * cos(theta2)
y2 =         L1 * sin(theta2)

d  = sqrt((x2-x1)^2 + (y2-y1)^2)
mx = (x1+x2)/2;  my = (y1+y2)/2
h  = sqrt(L2^2 - (d/2)^2)

# Upper solution
ex = mx - h*(y2-y1)/d
ey = my + h*(x2-x1)/d
```

Verification tests confirmed that FK(IK(x,y)) = (x,y) to within floating-point precision for all reachable workspace points. The five_bar_sim.py script provides an animated visualization of the arm moving

between target points, displaying both the FK-computed linkage geometry and the IK-computed joint angles in real time.

# Appendix B: Web Dashboard Interface

The OpenFPT web dashboard provides a visual interface for interacting with the test system. Built with vanilla HTML, CSS, and JavaScript (no framework dependencies), the dashboard features a dark-themed industrial design with amber accent colors and JetBrains Mono monospace typography, creating an aesthetic reminiscent of professional test equipment interfaces.

The dashboard provides the following functional panels: a File Upload panel with drag-and-drop support for KiCad .kicad_pcb and .kicad_sch files; a Board Overview panel displaying parsed component count, net count, and detected circuit patterns; a Test Plan panel showing the LLM-generated test sequence with probe coordinates and expected values; a Test Execution panel with real-time progress tracking as each probe command is dispatched; and a Diagnosis Report panel presenting the LLM fault analysis with color-coded severity indicators (critical/warning/info) and expandable fault detail cards.

The frontend communicates with the FastAPI backend via fetch() calls to the REST API endpoints. CORS is enabled to allow the dashboard to be served from any origin. The interface includes visual feedback elements such as animated scanline overlays, pulsing status indicators, and smooth CSS transitions for panel state changes. The design is fully responsive and works on both desktop and tablet screens.

The API server also auto-generates interactive Swagger UI documentation at the /docs endpoint, providing a complete interactive reference for all API endpoints with request/response schemas, example payloads, and direct testing capability.

> **Access the Dashboard:** Start the server with 'python pcb_api_server.py' and navigate to http://localhost:8000 in any modern web browser. For remote access during demonstrations, the pyngrok tunnel can expose the local server with a public HTTPS URL.