Kirin Autonomy, Inc.

Computer Science Field Session | Project Proposal

Fall Semester 2026 | 16-Week Engagement

Project	Autonomous Chess System: Board-to-Board Networked Play, a Calibrated Skill-Level System, and Web/Graphical Control
Client contact	Strydr Silverberg, Co-Founder | strydr@kirinautonomy.com
Preferred team size	3–4 students (4 preferred)
Location	Remote / on-campus, with scheduled access to physical hardware
Good fit for	Computer engineering students and anyone interested in hardware, embedded systems, robotics, networking, and full-stack software

Company Background

Kirin Autonomy, Inc. is an early-stage research and development startup founded by two Colorado School of Mines undergraduates, Strydr Silverberg and Colin Dake (Class of 2027). We build systems that tie software to physical hardware: software that doesn't just decide what to do, but actually goes and does it.

Our first product is an autonomous chess system. The system pairs a custom, from-scratch chess engine with a physically-actuated board: a two-axis gantry carrying an electromagnet moves the pieces, while a matrix of sensors detects the human player's moves. The result is a complete single-player, over-the-board chess experience against a computer opponent with no human opponent required, no screen between you and the pieces.

Kirin Autonomy is a **2026 Colorado School of Mines ProtoFund recipient**. The platform today includes:

- A bitboard chess engine with magic-bitboard move generation, Zobrist hashing, and alpha–beta search with iterative deepening, exposed over the standard UCI protocol.
- A hardware control stack (C++) driving a GRBL motion controller over serial, a 96-sensor hall-effect scanning array read through multiplexed GPIO, and an A* path planner that routes pieces around the board and into labeled capture-storage zones.
- An operator UI (I2C OLED status display plus physical start/stop/reset buttons) and an automated test suite (300+ assertions) wired into CI.
- A public-facing web application (React + Vite, deployed on Cloudflare) that hosts lab notes and project updates.

We are now ready to take Kirin from a working prototype to something we can put in front of people and stand behind, and that is where a Mines field session team comes in.

Description of the Work to Be Done

The team will take the chess platform we have today and make it playable across a network, give it a real range of difficulty settings, put it behind a browser, and build it a proper graphical front end. Testing is part of each of these rather than something saved for the end. The work splits into four workstreams, listed in the order they matter most to us. Four people can work on them in parallel; a smaller team would focus on the first three.

1. Play games over a streamed connection

Today Kirin is a single-player system: one person at one board. This workstream connects *two physical Kirin boards in different locations* so two people can play a live game against each other, each at their own board, with the pieces moving on both. When a player makes a move on their board (the hall-effect scanner picks it up), the other board's gantry plays that same move, and the other way around. Each person moves their own pieces by hand, and Kirin mirrors the opponent's moves onto the local board.

- A networked move channel that relays each player's detected move to the remote board, where the gantry executes it on the matching squares.
- Game-state synchronization between the two boards: a shared authoritative game state, legal-move validation, turn handling, and detection/recovery when a physical board and the digital state disagree (e.g., a misread sensor or a piece nudged out of place).
- Session handling: pairing two boards into a match, reconnecting and resyncing after a dropped connection, and getting captures, castling, en passant, and promotion right on both boards.
- An optional low-latency stream of each board so the players (and anyone watching) can see the opponent's physical board in real time.
- Testing as part of the work: harden the sensor and move-detection layer against noisy or unclear input, and build a hardware-in-the-loop or simulation harness that runs the whole pipeline (move detection, networked relay, path planning, G-code, simulated or real motion, sensor read-back), with regression tests that lock in correct behavior on both boards.

Likely technologies: WebSockets for move and state messaging, a small relay and matchmaking service, and WebRTC or something similar for the stream. Students aren't expected to know these on day one; picking them up is part of the project.

2. A complete, calibrated skill-level system

The engine has difficulty settings today, but they're coarse and we haven't checked them against any real rating. The team will build a fuller skill-level system that lets a player pick an opponent strength from **400 to 2200 Elo in 300-point steps** (400, 700, 1000, 1300, 1600, 1900, 2200). That covers everyone from a beginner still learning the rules to a strong club player.

- Ways to scale the engine's strength: search depth and time limits, some randomness and human-like blunders at the lower levels, and adjustments to the evaluation, so a given level feels like a real opponent at that rating instead of an engine that's been obviously held back.
- Checking the levels is the testing for this workstream: play batches of games at each setting

against opponents of a known rating (for example, bots or engines fixed at a target strength), look at the win, draw, and loss rates, and adjust until each level lands near the rating it claims. We'd want the results written down so the check can be repeated later.

- Make the chosen level settable through the engine's UCI interface, so it can be picked from the physical controls, the web interface (Workstream 3), and the graphical interface (Workstream 4).

3. Web interface

The team will build a browser-based control and monitoring interface for an operator, extending the existing React/Vite/Cloudflare stack:

- Live game view, move history, and current board state.
- Operator controls (start / stop / reset, skill-level selection, homing) that today live only on physical buttons and an OLED screen.
- A diagnostics dashboard surfacing raw sensor state, gantry status, and error conditions to make debugging the hardware far easier.
- Tests along with it: component and end-to-end tests for the interface and the link to the hardware, plus better CI so regressions get caught on their own.

4. Improved graphical interface

Right now you play through a terminal and the UCI interface. The team will design and build a graphical interface for gameplay: a rendered board, animated moves, a move list, an evaluation readout, and skill-level selection from Workstream 2. It should work both for single-player games on the board and as a companion view for the two-board networked matches in Workstream 1. As with the other workstreams, testing comes with it: coverage of the rendering and game-state logic, and regression tests to keep the interface in step with the engine.

Put together, these four pieces let two people in different places play a real over-the-board game against each other through their Kirin boards, let a player set the opponent's strength anywhere across a range we have actually checked, give an operator a browser dashboard to run and diagnose the hardware, and wrap it all in a graphical interface worth looking at. Because the testing is part of each piece, what comes out the other end is software we can stand behind rather than a one-off demo.

Desired Skill Set

We expect students to pick up new tools and languages during the course, so the list below is what helps, not a set of prerequisites. It's a good fit for computer engineering students and anyone curious about hardware, embedded systems, and robotics, and also for students who want solid full-stack and networking experience.

- **Most useful:** C/C++ (engine and hardware stack), JavaScript/TypeScript with a modern web framework (React), comfort on Linux, and Git.
- **Helpful:** networking and real-time streaming concepts (WebSockets, WebRTC); software testing

and CI; interest in embedded/hardware (serial, GPIO, sensors, motion control).

- **Mindset that matters most:** curiosity about systems that touch the physical world, willingness to debug across the hardware/software boundary, and eagerness to pick up unfamiliar tools.

Preferred Team Size

We'd prefer **four students** and can keep **three to four** busy. The four workstreams split cleanly across four people: say, two on streaming and networking and two on the skill-level system and the interfaces, with testing shared by everyone. A team of three would take on remote play, the skill-level system, and the web interface, and leave the more ambitious graphical work for later.

Location of Work

Work may be performed **remotely and on-campus**; no regular off-campus commute is required, which we know is important given fall-semester course loads. The Kirin hardware lives in the Golden area, and we will provide the team scheduled access to the physical hardware, both in person and remotely. Where a second board is needed to test the networked two-board play, we will coordinate access so the team can develop and demo it. Most development (web, GUI, networking, skill-level tuning, simulation-mode testing) can be done from anywhere; hardware-in-the-loop sessions will be coordinated around the team's availability.

Internship Potential

Yes. Kirin is an active, growing startup, and there's a real chance we'd offer one or more of the strongest contributors a continued role or internship after the field session. We'd be glad to keep working with students who want to keep building with us.

Non-Disclosure and Intellectual Property

Non-disclosure agreement: Students on this project will be asked to sign a non-disclosure agreement (NDA) before the engagement begins, covering non-public hardware, business, and product details.

Intellectual property: Students will be asked to assign intellectual property rights in the work and artifacts produced during the engagement to Kirin Autonomy, Inc. Students are encouraged to discuss their general experience and the technologies they learned; the NDA and IP assignment apply to the new, project-specific work and confidential materials.

We are excited to work with a Mines team and to give students a project where their software actually moves something in the real world.

Strydr Silverberg | Co-Founder, Kirin Autonomy, Inc. | strydr@kirinautonomy.com