Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Laidlaw Cohort 2025

Trinity College Dublin

Summer 1 Research Report

Blockchain Oracle-Based Reinforcement Learning for Dynamic Carbon Trading

Author: Shengxin Chen

Supervisor: Dr. Hitesh Tewari

September 2025

Abstract

This Summer 1 Research Reflective Report summarizes my work on building a prototype system while developing research and leadership skills set out in my Personal Development Plan (PDP). I adopted an incremental approach, starting small and using early feedback to guide each stage. Technical challenges—like system reliability, data validation, and adaptive decision-making—were tackled through iterative testing and clearer documentation. Alongside the technical progress, I learned to give concise updates, share prototypes early, and explain trade-offs clearly, which improved collaboration and decision-making. Looking ahead, I plan to extend the system with real-time data, explore multi-agent settings, and strengthen transparency to support future research and practical applications.

Introduction

Markets rarely behave in a predictable way. Prices rise and fall in rhythms that sometimes seem orderly and at other times chaotic. For carbon trading systems designed to incentivize decarbonization, this volatility is more than just an inconvenience. When prices collapse, industries lose the incentive to invest in cleaner technologies; when prices spike suddenly, firms face unexpected financial shocks. Either scenario clouds the signal the market is meant to send about long-term decarbonization goals.

This summer, I set out to explore whether modern digital technologies could help carbon markets behave in a steadier and more transparent way. The result was CarbonChainRL, a prototype system that combines three main elements: blockchain smart contracts, a data oracle, and a reinforcement learning (RL) agent. The blockchain holds the market's rules and records; the oracle brings reliable off-chain information on-chain; and the RL agent proposes small, measured adjustments to help stabilize the market when conditions change.

I began the project with two parallel goals. The technical goal was to build a functioning end-to-end prototype in which the learning agent suggests an intervention, the oracle validates the information, and the blockchain records both the decision and its rationale. The personal goal was to use this project as the basis for my Personal Development Plan (PDP): to plan and communicate my work more clearly, lead my own project responsibly, and treat setbacks not as failures but as data for learning.

Research Process and Methodology

The CarbonChainRL system was structured in four layers, each mapping onto a part of the user experience:

1. Presentation Layer: Dashboards for registration, auctions, and market monitoring to keep user interaction straightforward.
2. Business Logic Layer: Coordinates user actions with backend processes so the interface remains simple while the underlying system manages complexity.
3. Core Services Layer: Hosts the smart contracts, the oracle, and the RL agent. The contracts enforce the rules on-chain, the oracle verifies external data before writing it to the blockchain, and the learning agent suggests interventions based on market conditions.
4. Data and Storage Layer: Maintains historical records and model inputs, enabling both accountability and future analysis.

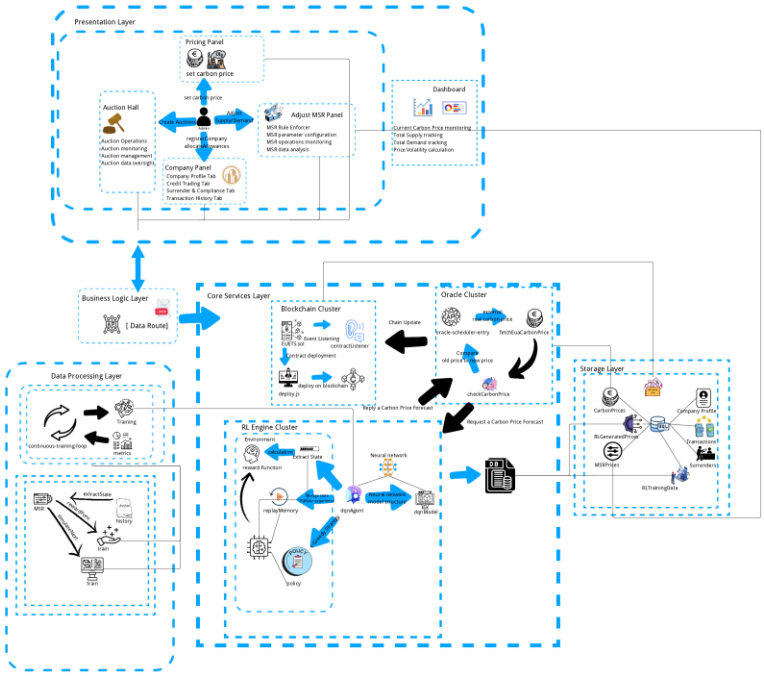

Figure 1. CarbonChainRL system architecture

On-chain, I developed a modular Solidity contract that implements core functionalities: sector-based registration, allowance allocation, peer-to-peer trading, configurable auctions, and a Market Stability Reserve (MSR) that releases or withholds allowances to smooth market volatility. When the market appears oversupplied, the MSR withholds allowances

from future auctions; when undersupplied, it releases additional allowances. Throughout development, I ensured that contract events were clear, parameters adjustable, and transactions auditable.

Off-chain, the RL agent observes a compact set of indicators—recent price movements, supply-demand balances, and MSR levels. Instead of trying to predict prices directly, the agent learns to make modest, targeted interventions aimed at keeping the system stable. The oracle sits between the two, acting as a gatekeeper. It verifies the reliability of incoming data, tags each update with a confidence score, filters anomalies, and only writes significant updates on-chain. This keeps transaction costs manageable while ensuring that each on-chain record carries an explanation.

Methodologically, I began with a full design on paper but quickly shifted toward incremental, vertical slices: deploying a minimal contract, connecting a simple oracle, training a basic RL model, then gradually expanding functionality. This approach revealed problems earlier, supported rapid iteration, and aligned well with short review cycles with my supervisor. Over time, moving from a single grand plan to iterative feedback loops proved to be one of the most valuable process changes of the entire project.

Challenges and Problem-Solving

The first serious issue was a learning agent that refused to learn. Runs looked random, then flat. The turning point came when I realised the reward was scaled so poorly that every action looked equally bad. In that world the agent was rational to give up. I rescaled the reward, added small unit tests around how it is computed, and logged what the agent believed it was seeing. Once failures left a clear trail, progress followed. The agent settled on smaller, sensible adjustments.

Figure 6: RL Reward Over Time with MSR Phase Context

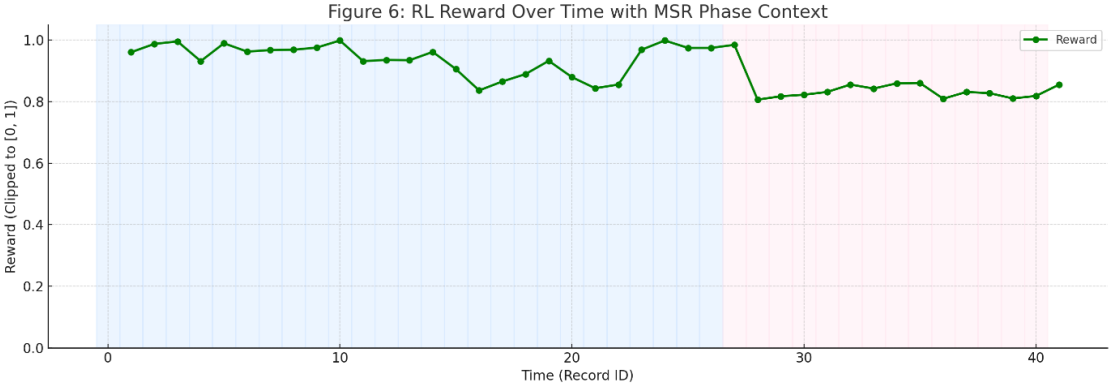

Figure 2. RL reward over time with MSR context

The second issue hid near thresholds. The contract behaved well in general, yet when indicators crossed a boundary that should have triggered an intervention, nothing happened. The fix was unglamorous but important. I tightened input checks, made events more descriptive, and wrapped multi step updates so they execute atomically. After that, even a rejected action explained itself. That change added more confidence than any new feature would have.

The third issue was rhythm. If the oracle writes too often, you pay in fees and noise. If it writes too rarely, the model's view goes stale. I added simple change detection and a confidence note so trivial or dubious updates stay off-chain and meaningful ones go through. The update pattern became easier to justify in plain language.

Across all three, one lesson stuck. Errors are not a referendum on competence. They are feedback about what the system still needs. Holding that view freed me to instrument the code and let it argue back with evidence.

Key Findings and Achievements

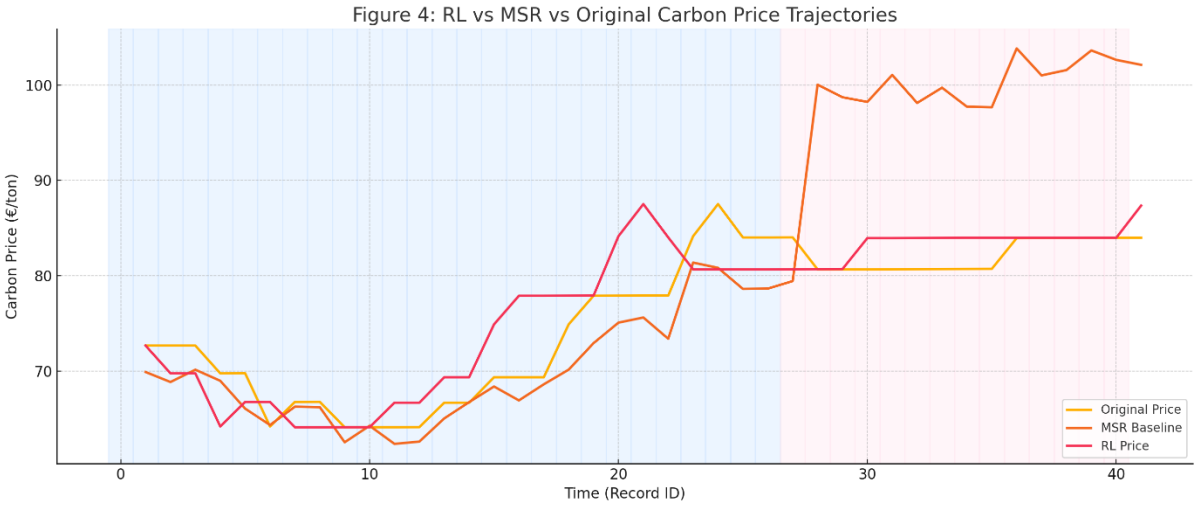

Figure 3. Price trajectories under a simulated shock

The development of CarbonChainRL successfully delivered a fully functional prototype integrating the reinforcement learning agent, the data oracle, and the blockchain smart contracts into a coherent end-to-end system. Under simulated price shocks, the learned policy produced smoother price trajectories compared to a fixed threshold-based intervention strategy, mitigating abrupt fluctuations while maintaining responsiveness. This

outcome demonstrates the potential of adaptive control mechanisms in stabilising carbon markets without compromising on intervention efficacy.

Relative to the initial project objectives, the primary technical milestones were achieved:

1. System Integration – The three core modules (reinforcement learning agent, oracle, and smart contracts) operated reliably within a single workflow.
2. Usable Front-End Interface – A functional dashboard was implemented for monitoring system performance and visualising interventions.
3. Academic Dissemination – The system design, methodology, and experimental results were documented in a conference paper submission.

However, several limitations were identified. First, most evaluations relied on validated or synthetic datasets rather than real-time market feeds, reflecting a deliberate choice to maintain oracle conservatism while system robustness was improved. Second, the complexity of implementing smart contract edge-case handling and atomic transaction logic was underestimated, resulting in extended development timelines. Future work should allocate explicit resources to observability, automated testing, and tooling, as these investments substantially improved reliability during later stages.

Two methodological insights emerged. The first was the value of incremental, end-to-end testing: achieving a minimal yet complete system cycle early in development accelerated debugging and system refinement. The second was the transition from feature expansion to feature consolidation—prioritising test coverage, explainability, and robustness before introducing additional complexity.

Personal and Leadership Development

Researcher Growth

I arrived attracted to elegant designs and left preferring systems that tell me plainly when and why they fail. Small experiments with clear success criteria beat sweeping plans. Writing tests for the reward felt tedious until I reframed it. The reward is the agent's compass. A broken compass guarantees a lost traveller. That simple image changed how I prioritised my time. I also learned to narrate my reasoning. When I could say what I expected and what would count as success, feedback improved and decisions got faster. A short practice

helped. After difficult tasks I wrote three lines: expectation, observation, adjustment. The habit reduced noise in my thinking and gave me language to explain my choices without defensiveness.

Leadership Lessons

During the research, the most helpful habit was opening discussions with a two minute brief. One sentence on the problem, the real options, and a recommendation with a simple reason. That turned status into choices and made it easier for others to respond. I also leaned into a collaborative, service oriented style. When integration stalled, the most valuable work was often to take on the unglamorous tasks no one owns: writing setup notes, preparing a reproducible demo, and adding minimal tests that keep regressions from creeping back.

A coaching session helped me name my strengths. I bring a growth focus, empathy, and relationship building. It also helped me name a gap I still feel. I can execute a plan, but persuasion is harder when plans change. The fix is not louder arguments. It is translation. I need to express technical trade offs as consequences that matter to people who do not share my background. I have started practising that and it is already helping.

An ethical thread ran through the build. A smooth looking policy is not a licence to act everywhere. Responsible leadership here means limits and explanations. Record why an intervention happened, show the data behind it, and keep a human override. Making the system understandable is part of making it safe.

Ethics & Responsibility

Regular check-ins mattered less for reporting and more for thinking better. Early reviews pushed me to turn a broad design into deliverable slices. Mid-project demos rewarded sharing early prototypes because issues surfaced while they were still cheap to fix. Later feedback on structure, figures, and references nudged me from code that runs to research that others can read and trust. Those interactions changed the project and changed me. I ask for feedback sooner, frame proposals as choices rather than updates, and document the reasoning so someone who was not there can still follow it.

Collaboration & Feedback

This work built directly on the four aims in my Personal Development Plan (PDP), turning them into habits I can carry forward. I moved from long progress updates to concise decision briefs with clear trade-offs and next steps, which improved discussions and made reasoning easy to revisit. Documenting key decisions reduced uncertainty for collaborators and helped when plans had to change, as implications could be explained to both technical and non-technical audiences.

Collaboration improved when I focused on enabling others: preparing clear setup instructions, adding lightweight tests for critical components, and running short, frequent prototype demos. These early demonstrations surfaced integration issues quickly and kept feedback specific and actionable. Protecting daily time for deep work while placing routine tasks outside it also gave the project a steadier rhythm.

Looking ahead, I plan to integrate external data sources while keeping the oracle conservative, encode policy context so interventions reflect genuine regulatory changes rather than noise, and strengthen on-chain auditability and role controls. From a research perspective, I will extend the system to multi-agent settings to test whether adaptive interventions remain effective when multiple actors interact. On the leadership side, I will continue using short briefs, early demos, and transparent decision records—that keep the project aligned and accountable.

Future Directions

Building on this foundation, future work will focus on three main directions:

1. **Integration of Real-Time Data**

The current system primarily relies on validated or synthetic datasets. Incorporating reliable, real-time data feeds will enable the oracle to handle genuine market signals while preserving its conservative, safety-oriented design.

2. **Policy Context and Multi-Agent Extensions**

Future iterations will encode explicit regulatory context so that interventions reflect authentic policy changes rather than noise. Moving from a single learning agent to multi-

agent environments will allow testing whether adaptive interventions remain effective when multiple actors respond strategically to one another.

3. **Enhanced Transparency and Auditability**

Strengthening on-chain role controls, decision logs, and intervention explainability will ensure the system remains accountable to both technical and non-technical stakeholders, supporting real-world deployment and regulatory oversight.

Beyond technical improvements, leadership practices such as short decision briefs, early demonstrations, and clear trade-off communication will remain central to keeping the project aligned, transparent, and open to scrutiny as complexity grows.

Conclusion

I hoped for neat convergence. I got something better, which was a system I could learn from. Breaking work into pieces I could finish, insisting on observability, and writing down what I thought would happen before I tested it made my work sturdier and my decisions calmer. Leadership turned out to be less about title and more about turning ambiguity into choices, inviting critique, and taking responsibility for clarity and ethics.

The result is a working prototype rather than a product. It is, however, a credible proof of concept that shows how a learning agent, a careful oracle, and transparent contracts can act together without becoming a black box. That justifies the next stage. More importantly, the summer changed how I work. I am more comfortable sharing early prototypes, more willing to try small experiments, and more patient with slow but important craft like writing tests and tightening edge cases. These habits will outlast this project and shape how I lead and learn in the next phase of the programme.