1. Project Setup & Organization

Set up project management tools (Jira/Atlassian)

Create tasks, subtasks, and milestones.
Establish sprint planning for weekly or bi-weekly review.

Document the high-level project vision

Write a project overview.
Define core objectives: analog signal processing, adaptive gates, signal language.

Plan basic cybersecurity and safety measures

Define initial concerns for privacy and security in analog-binary hybrid system.
Begin outlining how system should handle external threats or malfunctions.

2. Signal Language Design

Research and define signal components

Break down signal into amplitude, frequency, phase, and time properties.
Determine how analog and binary data will be embedded or layered within the signal.

Develop modulation rules for signals

Define how each signal component (amplitude, phase, etc.) will modulate for different types of operations.
Consider how the signal will scale for different processing requirements.

Design signal language encoding

Develop a language to encode data and instructions into the signal.
Plan transitions between binary and analog forms in processing.

3. Gate Design & Research

Research gate components

Review component options: op-amps, transistors, resistors, capacitors, etc.
Explore how transistors or op-amps could support both analog and binary logic operations.

Plan gate functionality and signal modulation

Define initial gate prototypes (e.g., NAND, OR).
Explore how gates can process both binary and analog signals, potentially alternating between modes.

Design adaptive gates

Consider designs for gates that change functionality based on signal inputs (adaptive nodes).
Investigate how feedback from the system could trigger a change in the gate's role or behavior.

4. Hardware Prototyping

Shop for additional components

Make a list of all needed parts for the breadboard prototype (transistors, resistors, op-amps, capacitors, etc.).
Purchase the necessary components for building the initial prototype.

Set up initial breadboard test environment

Build the first gate (start with NAND gate).
Test initial signal processing (analog and binary modes).
Set up power supply and measurement tools.

Begin exploring feedback mechanisms

Develop simple feedback loop experiments to observe how the system handles error correction.
Explore how feedback can assist with gate adaptation and learning.

5. System Behavior and Learning

Design node-to-node interaction logic

Develop the interaction rules for nodes, with emphasis on "help your neighbor" principles.
Limit depth of interactions as a safeguard to avoid runaway processes.

Plan calibration/training tasks

Create simple calibration routines for the system to handle through the ALU initially.
Consider how training could eventually be passed on to the adaptive gates.

Define memory storage methods

Research analog equivalents of data storage (e.g., capacitors).
Plan how the system will handle short-term and long-term memory.
Begin outlining data storage mechanisms for nodes and gates.

6. Testing & Simulation

Test gate functionality

Run initial tests for the gates built on the breadboard.
Observe how the gates handle signal modulation and logic operations.

Design error correction and self-healing tests

Develop protocols for error correction using feedback loops.
Implement tests to observe how gates and nodes self-heal in the event of signal loss or failure.

Run larger simulations for system behavior

Plan initial simulations for a decentralized network of nodes.
Test interactions between nodes and gates, focusing on how they work together to process signals and maintain system stability.

7. Documentation & Continuous Review

Document all processes and outcomes

Keep thorough documentation on signal modulation, gate design, feedback loops, and node interaction.
Review progress weekly to update project scope or reprioritize tasks.

Plan iterative reviews

Set review points for design refinements, based on feedback from tests and simulations.
Adjust signal language and gate design based on results from each iteration.