Your approach makes sense—starting with a hybrid system that emulates a binary computer using analog components could give you a solid foundation before diving deeper into the potential of analog logic. This would allow you to leverage your current understanding of binary computing while gradually introducing and experimenting with analog systems.

Here's how you might break down actionable steps:

1. Emulate Binary Computing with Analog Logic Gates

Design Hybrid Gates: Develop analog logic gates (e.g., NAND, NOR) that mimic binary gate operations. These gates should accept continuous input but output binary-like decisions.
Power Modulation for Input/Output: Use the energy signatures from analog power modulation to represent binary states. Start simple by modulating voltage or current to replicate `0` and `1` signals.

2. Build a Simple Emulated Circuit

Simple Arithmetic/Logic Unit (ALU): Build an ALU that can perform basic binary operations (addition, subtraction, etc.), but designed with analog gates. This should give you a functional understanding of how analog circuits could mimic traditional binary computing.

3. Analog Emulation Layer

Binary-Analog Translator: Create a system that translates binary instructions into the appropriate analog signatures (and vice versa). This will serve as the interpreter between the binary and analog domains.
Execute Simple Binary Programs: Write and run simple programs that execute within this hybrid environment to confirm that your analog system can emulate binary processes.

4. Explore Native Analog Operations

Develop Analog Logic: Once the binary-emulating layer is functioning, start experimenting with pure analog operations. These would include logic that doesn't rely on the strict binary constraints but uses continuous values.
Run Analog Programs: Create and execute analog-based programs within the system, possibly running these within the binary emulator layer for simplicity and comparison.

5. Test Real-World Applicability

Run Benchmarks: Compare the performance of analog emulated programs versus traditional binary programs. This could give insights into where analog computing could excel or fall short.
Explore Hybrid Programs: Begin creating programs that take advantage of both the analog and binary nature of the system.

6. Scalability and Future Prototypes

System Scalability: Once the initial prototype is validated, explore how to scale the system for larger, more complex tasks.
Full Analog Systems: After becoming familiar with analog logic, explore building a full analog system that doesn't require binary emulation, experimenting with entirely new computational models.

This roadmap should give you actionable steps to move forward while keeping the process familiar at first before moving into uncharted territory.

How does that structure sound for you as a plan?