ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Boost Converter Design Agent based on PE-GPT and PANN

Team #23

JIARONG XU
HAOJUN LI
XINYU ZHAN
JIAMING MA

TA: Yujie Gong

May 2026

Abstract

This project develops a Boost Converter Design Agent based on PE-GPT and PANN for power-electronics design, verification, and diagnosis. The agent follows a Brain-Planning-Tool architecture in which the LLM brain interprets requirements, retrieves domain knowledge, uses planning to decompose requests into executable steps, and calls tools for physical-model design, closed-loop hardware control, measurement, and PANN diagnosis. Given voltage targets, load, switching frequency, and nonideal terms, the agent designs duty cycle, inductor, and output capacitor using a nonideal continuous-conduction-mode (CCM) boost model. PCB measurements are compared with predicted behavior. When diagnosis is requested, the agent processes uploaded data files with the PANN model to estimate parasitic parameters, analyze mismatch sources, and suggest design improvements. Although the hardware demonstration focuses on an asynchronous boost converter using a GaN half-bridge module, the same planning structure can extend to RLC networks and buck converters.

Contents

1 Introduction

Power electronics design usually requires both analytical circuit knowledge and practical iteration on real hardware. Even familiar circuits become difficult for a new designer once topology selection, nonideal components, parasitic resistance, measurement uncertainty, and controller settings are included. The goal of the Boost Converter Design Agent based on PE-GPT and PANN is to make that design process more interactive and reliable by combining a conversational interface with physics-based design tools and hardware feedback.

1.1 Problem Statement

Power-electronics design is difficult to automate because a useful answer must connect circuit requirements, topology selection, nonideal device behavior, component availability, and hardware measurements. A simple calculator can return a duty cycle or passive value, but it does not explain whether the selected topology is appropriate, which assumptions were used, or why measured hardware may disagree with the prediction. This report uses an asynchronous boost converter implemented around a GaN half-bridge module as the main hardware case. The converter steps an input voltage up to a higher output voltage while operating in continuous conduction mode (CCM). In an ideal classroom model, the voltage gain is written as

$$V_{out} = \frac{V_{in}}{1 - D}, \tag{1}$$

where D is the duty cycle. The hardware used in this project is not ideal. A more realistic topology includes the SS34 input-path drop, the GaN body-diode drop during reverse conduction, and the lumped path resistance. Using the duty ratio to combine the switching intervals, the demonstrated converter is described by

$$L\frac{di_L}{dt} = V_{in} - V_{SS34} - (1 - D)(V_{out} + V_{d,body}) - i_L R_{total}, \tag{2}$$

$$C_{out}\frac{dV_{out}}{dt} = (1 - D)i_L - \frac{V_{out}}{R_0}. \tag{3}$$

These equations explain why the calculated component values and controller settings can differ noticeably from the values required on the actual PCB. In practice, however, the exact diode drops, lumped resistance, and installed passive values are difficult to know before measurement. This uncertainty motivates the measurement feedback and PANN diagnosis used later in the workflow.

1.2 Project Objective

The objective is to build an LLM-centered circuit-design agent that can translate user requests, autonomously plan the design process, and guide the user through circuit-design tasks. After reading a user's request, PE-GPT should interpret the selected circuit type,

identify the relevant constraints, organize an appropriate sequence of steps, and call the tools needed for that plan. In the boost hardware demonstration, the agent explains the selected asynchronous GaN boost topology, collects the design requirements, uses a non-ideal physical model tool to design the duty cycle and passive components, sends the duty command to the PCB, compares measured and predicted behavior, and calls the PANN model for diagnosis when requested.

The demonstrated hardware path is a boost converter, but the agent structure is not limited to a single hard-coded boost calculator. Other circuit types can be supported by adding their own requirement schema, design procedures, physical models or trained models, output parameters, and verification checks.

1.3 Report Organization and Main Findings

The remainder of this report follows the agent design and verification structure. Chapter 2 presents the design of the PE-GPT agent, the asynchronous boost power stage, the nonideal physical model, the closed-loop hardware path, and the PANN diagnosis function. Chapter 3 describes the verification plan, including the design-tool checks, controller-command path, hardware-measurement placeholders, and PANN diagnosis requirements. Chapter 4 summarizes cost, schedule, safety, and ethical considerations. Chapter 5 concludes that the project is best understood as an LLM-centered circuit-design agent: PE-GPT interprets user requirements, plans the task, and calls physical-model, hardware-control, and PANN tools as needed. The demonstrated hardware path is a boost converter, while the agent structure can be extended to other circuit types through additional tools and design steps.

2 Design

2.1 Hybrid Interaction Design

The interface supports both structured parameter input and natural-language interaction. Structured forms are stable and suitable for numerical parameters, while natural-language input allows the user to describe the design target more flexibly.

For example, the user may provide a request such as:

> "Design an asynchronous boost converter from 12 V to 24 V with a switching frequency of 100 kHz."

The parser extracts the circuit type and corresponding physical quantities from the sentence. For the asynchronous boost-converter workflow, the extracted parameters include input voltage, target output voltage, load resistance, switching frequency, SS34 voltage drop, body-diode voltage drop, and lumped parasitic resistance.

The hybrid interaction structure is important because it combines flexibility with stability. Natural-language input improves usability and makes the assistant appear more intelligent, while the structured parameter interface ensures that the downstream physical model receives consistent numerical input.

The interaction process also includes retry and confirmation stages. The user may revise the generated workflow, correct invalid parameters, or confirm hardware preparation before continuing. This staged interaction makes the system behave more like a collaborative engineering assistant.

2.2 Agent Planning and Workflow Design

Traditional power-electronics design tools are usually implemented as static equation calculators. Although such tools can generate numerical parameters, they typically cannot guide the user through topology understanding, requirement validation, staged reasoning, hardware preparation, or diagnosis. In practical converter design, users often do not initially know which parameters are required, which assumptions are physically meaningful, or how nonideal effects influence the final hardware behavior. Therefore, the PE-GPT framework redesigns the interaction process as an LLM-centered engineering workflow assistant rather than a standalone calculation script.

Instead of forcing the user into a completely fixed sequence, the assistant dynamically organizes the workflow according to the design request and current interaction state. The LLM first interprets the target circuit and then generates a planning-oriented reasoning process before numerical calculation begins. This design allows the interface to behave more like an engineering assistant that guides a design task progressively rather than a traditional form-based calculator.

2.2.1 Motivation of Guided Workflow

The main motivation for introducing a guided workflow is that practical converter design is usually incomplete and iterative. Users may initially provide only partial information such as target voltage or switching frequency. In many cases, the user may not know whether the requested operating condition is physically reasonable or whether additional nonideal parameters should be considered.

A conventional calculator assumes that all required quantities are already known. By contrast, the proposed workflow attempts to reduce the engineering burden on the user. The assistant first identifies the selected topology, explains the expected design path, requests missing parameters, validates the operating condition, and only then activates the design tool.

The workflow structure also improves interpretability. Instead of directly returning final values, the assistant exposes intermediate reasoning stages such as topology understanding, planning generation, physics assumptions, and hardware preparation. This staged interaction helps users understand not only what the result is, but also why the result is produced.

In addition, the workflow improves operational safety. For example, in the boost-converter branch, the assistant checks whether $V_{out} > V_{in}$ before activating the sizing tool. If the operating condition is inconsistent with boost-converter physics, the workflow pauses and requests correction instead of continuing with invalid assumptions.

2.2.2 LLM-Centered Planning

The AI planning stage is placed immediately after topology selection. Once the user selects or describes a circuit type, the LLM generates a suggested engineering workflow according to the requested task. The workflow is therefore treated as planning-oriented reasoning rather than a hard-coded execution script.

For the asynchronous boost-converter demonstration, the generated plan usually includes understanding the topology, collecting parameters, validating the request, running the physical model tool, preparing hardware instructions, comparing predicted and measured behavior, and activating diagnosis when mismatch becomes significant.

This planning stage is important because it changes the role of the LLM from a text generator into a workflow coordinator. The assistant decides which stage should appear next and which tool should be activated according to the current design state. The generated workflow therefore behaves more like an AI-guided engineering process than a static interface.

The planning stage also improves the interaction experience. Instead of immediately showing every design stage at the beginning, the workflow is progressively expanded after the user confirms the generated plan. This design avoids exposing the entire process as a pre-written pipeline before planning has been completed.

2.2.3 Workflow State Management

The assistant maintains an internal workflow state throughout the interaction process. The workflow state includes the selected topology, validated parameters, active design stage, hardware confirmation status, and diagnosis readiness.

The interface displays workflow progress in a sidebar. At the beginning, only topology selection and AI planning are visible. After the planning stage is confirmed, additional stages become available progressively. This design prevents the workflow from appearing as a completely fixed sequence before planning has been completed.

The workflow state also allows the assistant to preserve context between stages. Instead of treating each message independently, the assistant remembers previous design assumptions and uses them to determine the next engineering action. This behavior makes the system closer to a real engineering workflow manager.

Another advantage of workflow-state management is error handling. When invalid parameters are detected, the assistant pauses the workflow and highlights the problematic quantities instead of silently continuing. Corrected inputs can then be inserted directly into the current workflow stage without resetting the entire process.

2.2.4 Multi-Topology Extensibility

The current interface contains three topology branches:

- asynchronous boost converter;
- buck converter;
- RLC oscillator.

The asynchronous boost converter is the most complete implementation and includes planning, validation, physical-model design, hardware preparation, and diagnosis stages. The buck-converter and RLC-oscillator branches currently share the same planning and interaction framework, but their detailed physical tools are left for future work.

This structure demonstrates that the system is not implemented as a hard-coded boost-converter calculator. Instead, the workflow architecture is designed to support extensibility. Different topologies can reuse the same planning structure while attaching different physical models, parameter schemas, validation rules, and diagnosis procedures.

The extensible structure is important because power-electronics systems often involve many converter types with different physical behavior. By separating planning logic from topology-specific tools, the framework can be expanded without redesigning the entire interaction process.

The same agent interface can support different circuit types if each type provides its own required inputs, design procedure, physical model or trained model, output parameters, and verification checks. In this report, the implemented hardware path is the

asynchronous GaN boost demonstration. The detailed boost workflow is summarized below.

1. select the asynchronous GaN boost-converter type;
2. explain the selected topology, CCM operation, and the nonideal voltage-drop assumptions;
3. collect design requirements;
4. check whether the requested input voltage, output voltage, and switching frequency are reasonable;
5. set initial design assumptions such as inductor-current ripple and output-voltage ripple;
6. call a nonideal physical model tool to compute recommended duty cycle, inductance, and capacitance values;
7. call the hardware-control interface to send the recommended duty cycle to the PCB control path;
8. guide the user to install or confirm the recommended passive components and enable the driver supply;
9. read measured output voltage and inductor current from the hardware after the user confirms that the hardware setup is complete;
10. call the prediction tool with the designed duty cycle, inductance, and capacitance to calculate predicted output voltage and inductor current, compare them with measured results, and give a brief error analysis;
11. ask the user to upload processed data files collected under different input-voltage conditions when diagnosis is requested;
12. call the PANN tool to estimate hidden hardware parameters such as the actual inductance, capacitance, lumped resistance, and body-diode drop;
13. use the estimated parameters and measurement comparison to analyze mismatch sources;
14. suggest design improvements based on the diagnosis results; and
15. guide the user back to hardware-circuit verification after the suggested improvements are applied.

Figure 1 summarizes this sequence as a workflow visual aid for the boost hardware demonstration. It is intended to show the order of interaction and tool use rather than a static module relationship.

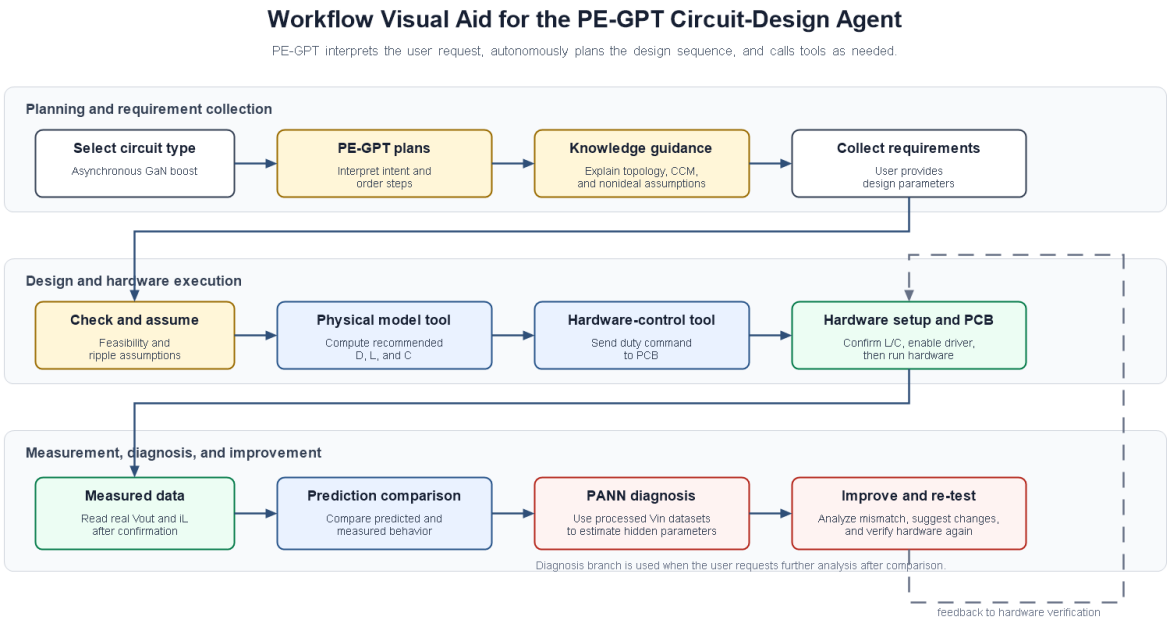

Figure 1: Workflow visual aid for the PE-GPT circuit-design agent and boost hardware demonstration.

The workflow in Figure 1 is representative. The LLM may shorten, reorder, or extend the plan depending on the user's request, the selected circuit type, the available measurements, and whether the user asks for diagnosis.

2.3 Physical Model and Design Tool

The physical model has two roles in the final system. First, it explains the boost-converter physics used for the hardware demonstration. Second, the same model is implemented as a callable PE-GPT tool that returns design values and prediction results. The model begins with the ideal relationship between input voltage, output voltage, and duty cycle, then adds dominant nonideal terms identified for the actual circuit [1].

For an ideal boost converter in CCM, volt-second balance across the inductor gives

$$DV_{\text{in}} + (1 - D)(V_{\text{in}} - V_{\text{out}}) = 0. \tag{4}$$

Solving for the ideal duty cycle gives

$$D_{\text{ideal}} = 1 - \frac{V_{\text{in}}}{V_{\text{out}}}. \tag{5}$$

This equation is useful for explanation, but it is not sufficient for final hardware because it assumes zero diode drop and zero parasitic resistance. At steady state, the approximate nonideal output voltage is

$$V_{\text{out}} \approx \frac{V_{\text{in}} - V_{\text{SS34}} - I_L R_{\text{total}}}{1 - D} - V_{d,\text{body}}. \tag{6}$$

For the boost demonstration, the design tool uses fixed initial ripple assumptions,

$$\frac{\Delta I_L}{I_L} = 0.2, \qquad \frac{\Delta V_{\text{out}}}{V_{\text{out}}} = 0.01. \tag{7}$$

For an initial sizing pass, the output current is estimated as

$$I_{\text{out}} = \frac{V_{\text{out}}}{R_{\text{load}}}, \tag{8}$$

the inductor is sized from the switch-on inductor voltage,

$$L \approx \frac{(V_{\text{in}} - V_{\text{SS34}} - I_L R_{\text{total}})D}{f_s \Delta I_L}, \tag{9}$$

and the output capacitor is sized from the expected output ripple,

$$C \approx \frac{I_{\text{out}} D}{f_s \Delta V_{\text{out}}}. \tag{10}$$

The PE-GPT tool wrapper exposes this calculation through a compact input schema: V_{in}, V_{out}, R_{load}, f_s, V_{SS34}, $V_{d,\text{body}}$, and R_{total}.

For one example requirement set used during workflow development, the design tool returned the values shown in Table 1. The nearest available hardware modules from the project inventory are the 1 mH inductor and 220 μF output capacitor.

Table 1: Example recommended design values from the updated tool.

Quantity	Recommended value
Duty cycle D	0.500
Inductance L	720.00 μH
Output capacitance C_{out}	217.01 μF

2.4 Hardware Circuit and Closed-Loop Control

The hardware demonstration uses an asynchronous boost converter built around five main hardware blocks: a WeAct GD32F303CCT6 control board, a GSP65R13HB-EVB GaN half-bridge board, a voltage/current sampling module, a CH340 USB-to-serial module, and a custom PCB that carries the remaining boost-converter circuit and interconnections. The custom PCB provides the passive-component and load connection path, while the GaN half-bridge board provides the switching power stage. The hardware path is treated as a closed-loop experimental path because PE-GPT does not stop after producing component values. It sends the duty command to the controller, waits for the user to confirm the hardware setup, reads measurements, and uses the measured data in later comparison and diagnosis steps.

The controller-side implementation uses a WeAct GD32F303CCT6 board as the PWM and data-acquisition controller. We first programmed the GD32 board with embedded C firmware that initializes the PWM timer, UART interface, ADC channels, and enable pin. A CH340 USB-to-serial module connects the computer to the GD32 UART interface. The CH340 TXD pin is connected to PA10, which is the GD32 USART0 receiver, and the CH340 RXD pin is connected to PA9, which is the GD32 USART0 transmitter. The CH340 module also shares 3.3 V and ground with the GD32 board. The sampling module sends its current-sensing output I_{os} to PA0 and its voltage-sensing output V_{os} to PA1, while sharing ground with the controller. The sampling module requires an independent 5 V supply, so it is not powered from the GD32 3.3 V rail.

The serial interface runs at 115200 baud and gives PE-GPT a simple hardware-control layer. On the agent side, the hardware interface is implemented as a Python control tool using `pyserial`. This tool opens the CH340 serial port, sends text commands to the GD32 firmware, parses the returned lines, and exposes functions such as duty-cycle setting, sensor reading, PWM start/stop, and status query to the PE-GPT workflow. In this structure, PE-GPT can call the Python functions as hardware-control tools and use the returned measurement values to decide the next workflow step. Table 2 summarizes the command interface used in the hardware workflow.

Table 2: Serial command interface for the closed-loop hardware path.

Command	Function	Example response
`D 0.55`	Set PWM duty cycle	`OK: duty=0.5500 pulse=1320/2400`
`R`	Read calibrated voltage and current measurements from the sampling path	`PHY: Io=0.4610A Vo=20.670V`
`C`	Calibrate the measurement zero offset with power off	`CALIB: ios_zero=1.4362V`
`S`	Stop PWM and disable the output stage	`OK: PWM stopped, EN=LOW`
`G`	Start PWM and enable the output stage	`OK: PWM started, EN=HIGH`
`?`	Query controller status	`Status: PWM=ON duty=0.5500 EN=HIGH`

Because the sampling module and ADC path have limited precision, the measurement software does not rely on a single ADC conversion. In the GD32 firmware, each ADC read averages 256 ADC conversions before returning a 12-bit value. This reduces the effect of switching-ripple phase and random ADC noise in the sampled voltage and current signals. The firmware also includes a zero-offset calibration command. When the input supply and driver supply are turned off, the `C` command records the sensor zero offset by averaging 64 firmware-side ADC readings. The reported measurement values are then computed from the sampled ADC voltages after subtracting the calibrated offset where needed.

The agent-side serial-control program adds a second averaging layer. For a normal measurement read, the program sends the `R` command 50 times with a 50 ms interval between readings. It then applies a 3-sigma outlier filter to remove abnormal samples and reports the average and standard deviation of the remaining voltage and current measurements. For zero calibration, the program first triggers the firmware-side 64-sample calibration and then verifies the zero point with 50 additional serial readings. These averaging and filtering steps make the serial measurement more stable before the values are passed back to PE-GPT.

After D, L, and C are recommended by the physical model tool, the agent sends the recommended duty cycle to the GD32 through this serial interface. The user then installs or confirms the recommended passive components and enables the driver power supply. Once the user confirms that the hardware setup is complete, the agent can start PWM, read the measured voltage and current signals from the sampling path, and stop PWM

when needed. These measurements are returned to PE-GPT so that the agent can compare the physical circuit with the model prediction. In this sense, the loop is closed at the experiment-control level: model-based design values are applied to real hardware, hardware responses are measured, and the measured data are passed back to the agent for comparison and diagnosis.

2.5 PANN-Based Diagnosis

When the user requests diagnosis, PE-GPT asks for processed data files collected under different V_{in} values for the same hardware circuit. The agent then calls the PANN tool to estimate hidden hardware parameters such as the actual L, C, R_{total}, and $V_{d,body}$. This result helps explain why the original design may have missed the target output voltage or current.

The PANN model follows the physics-in-architecture idea: the learnable parameters correspond to circuit quantities rather than arbitrary neural-network weights [2]. In the implemented diagnostic tool, the learnable quantities are stored in logarithmic form so that the recovered physical parameters remain positive during optimization. The estimated parameter set includes L_0, C_{out}, R_o, R_{total}, and $V_{d,body}$. The model uses the same nonideal boost-converter structure as the physical tool, with measured V_{in}, switching state or duty ratio, inductor current, and output voltage as data inputs:

$$\frac{di_L}{dt} = \frac{V_{in} - (1-D)(V_{out} + V_{d,body}) - i_L R_{total}}{L_0}, \tag{11}$$

$$\frac{dV_{out}}{dt} = \frac{(1-D)i_L - V_{out}/R_o}{C_{out}}. \tag{12}$$

For raw switching data, D is the measured switch state 0 or 1. For averaged data, D is the period-averaged duty ratio.

The uploaded CSV files are processed at two time scales. At the raw scale, the tool uses the measured columns Time, V_{in}, V_{out}, i_L, switching state, and load resistance to build Euler rollouts at the original 80 ns sampling interval. Each raw rollout contains 250 steps, corresponding to one 20 μs switching period. At the averaged scale, the script groups every 250 raw samples into one period-averaged point and builds 50-step windows, corresponding to about 1 ms. The averaged windows are automatically selected between 4 ms and 30 ms using observability and noise criteria, including minimum output-voltage change, bounded duty variation, bounded output-voltage step size, and positive estimated charge transfer.

Training is performed jointly on the raw and averaged datasets. The raw-scale loss anchors the inductor-current rollout. The averaged-scale loss anchors the output-voltage window residual. The averaged voltage residual uses the exact period-averaged capacitor current term $\langle(1-sw)i_L\rangle$, which better reflects the charge delivered to the output capacitor over a switching period. The optimizer updates the physical parameters with Adam, uses gradient clipping, and selects the final model according to the lowest independent group-test loss. In the current implementation, part of the uploaded data files is used

for training, while the remaining files are reserved as an independent test group to check whether the identified parameters generalize beyond the training operating points.

After the PANN tool returns the estimated parameters, PE-GPT compares them with the nominal design values and the earlier prediction error. The agent can then identify likely mismatch sources, such as inaccurate parasitic resistance, body-diode voltage assumptions, load resistance, or passive-component values, and suggest design improvements. After those improvements are applied, the user can return to the same hardware path described above for another verification run.

3 Design Verification

This section verifies the final PE-GPT boost-converter design agent according to the actual workflow implemented in the project. The goal is not only to check whether the boost converter can reach a target output voltage, but also to check whether the agent can interpret a user request, plan the design process, call the correct tools, command the hardware path, compare measurements with the physical model, and use PANN for diagnosis when the measured circuit differs from the prediction.

The final workflow is verified as an integrated Brain-Planning-Tool system. PE-GPT is responsible for requirement interpretation and task planning. The nonideal boost model is responsible for forward design and prediction. The hardware-control path is responsible for applying the selected duty cycle to the PCB. The PANN model is used only in diagnosis mode to estimate hidden hardware parameters from measurement data. This division is important because the known converter physics should remain interpretable, while machine learning is used only where the physical parameters cannot be directly observed.

3.1 Verification Strategy

Verification is organized around the same sequence used by the agent during a normal boost-converter design run. First, the user selects the asynchronous GaN boost-converter path. The agent must then collect the operating requirements and the nonideal parameters of the circuit, including the input voltage, target output voltage, load resistance, switching frequency, SS34 diode drop, GaN body-diode drop, and lumped path resistance. After the requirements are collected, the physical-model tool computes the duty cycle and passive component values. The hardware-control interface then sends the duty-cycle command to the controller, and the measured output voltage and inductor current are compared with model predictions. If the error is too large, the agent asks for processed datasets and calls the PANN diagnosis tool.

The final verification requirements are summarized in Table 3. This table replaces the earlier proposal-style verification list with requirements that match the final project design.

3.2 Physical-Model Verification

The physical-model tool is verified first because it is the forward design engine of the final system. The tool input schema is

$$V_{\text{in}}, \quad V_{\text{out}}, \quad R_{\text{load}}, \quad f_s, \quad V_{\text{SS34}}, \quad V_{d,\text{body}}, \quad R_{\text{total}}. \tag{13}$$

The duty cycle must satisfy

$$0 < D < 1. \tag{14}$$

A returned value outside this range would indicate that the requested operating point is not achievable by the selected boost path or that the tool call was invalid.

Table 3: Verification plan for the final PE-GPT boost-converter workflow.

Function	**Verification method**	**Pass criterion**
Agent planning	Select the asynchronous GaN boost path and provide a compact design request.	PE-GPT requests the required electrical parameters and nonideal terms before calling tools.
Physical-model design	Run the nonideal boost design tool using the development requirement set.	The tool returns a valid duty cycle $0 < D < 1$ and passive values that can be mapped to available components.
Hardware command	Send the returned duty cycle to the PCB control path.	The PWM duty cycle and switching frequency match the command before the converter is energized.
Measurement comparison	Measure V_{out} and i_L from the PCB and compare them with the model prediction.	Voltage and current errors are within the selected tolerance, or the agent correctly triggers diagnosis.
PANN diagnosis	Upload processed data collected under multiple input-voltage conditions.	The PANN tool returns physically plausible estimates of L, C, R_{total}, $V_{d,\text{body}}$, and V_{SS34}.

The model is based on the averaged nonideal CCM boost-converter equations [1]:

$$L\frac{di_L}{dt} = V_{\text{in}} - V_{\text{SS34}} - (1-D)\left(V_{\text{out}} + V_{d,\text{body}}\right) - i_L R_{\text{total}}, \tag{15}$$

$$C_{\text{out}}\frac{dV_{\text{out}}}{dt} = (1-D)i_L - \frac{V_{\text{out}}}{R_{\text{load}}}. \tag{16}$$

At steady state, the output-voltage estimate becomes

$$V_{\text{out}} \approx \frac{V_{\text{in}} - V_{\text{SS34}} - I_L R_{\text{total}}}{1-D} - V_{d,\text{body}}. \tag{17}$$

Equation 17 is the main verification check for the physical model because it shows that diode drops and parasitic resistance directly affect the duty cycle required to reach the target voltage. Therefore, the design tool is considered valid only if it uses the nonideal terms instead of applying the ideal boost equation alone.

For the development requirement set, the physical-model tool returned the values shown in Table 4. The calculated capacitance is close to the available $220\ \mu\text{F}$ capacitor, while the calculated inductance can be implemented using the nearest available $1\ \text{mH}$ inductor. This verifies that the tool output can be translated into real prototype components instead of remaining only a mathematical result.

Table 4: Representative physical-model output.

Quantity	**Tool output or selected value**
Duty cycle D	0.500
Calculated inductance L	$720.00\ \mu\mathrm{H}$
Selected inductor	$1\ \mathrm{mH}$ module from inventory
Calculated output capacitance C	$217.01\ \mu\mathrm{F}$
Selected output capacitor	$220\ \mu\mathrm{F}$ module from inventory

3.3 Hardware-Control and Measurement Verification

After the model returns the recommended duty cycle, the next verification item is the command path from PE-GPT to the PCB controller. The duty-cycle command should be checked before the relay is closed or the converter is fully energized. This prevents an incorrect PWM command from overstressing the inductor, capacitor, GaN device, or load resistor. For the development case, the command is $D = 0.500$, so the expected PWM waveform should have an approximately 50% duty cycle at the selected switching frequency.

The hardware-control verification procedure is as follows:

1. Send the calculated duty cycle D from PE-GPT to the controller.
2. Observe the PWM output using an oscilloscope or logic analyzer.
3. Confirm that the measured duty cycle and frequency match the command.
4. Install or confirm the recommended passive components.
5. Enable the driver supply and run the converter only after the PWM waveform is verified.

The six uploaded switching datasets were then processed to fill the measurement table. For each dataset, the final 10% of the waveform was averaged to represent steady-state behavior. The prediction columns were calculated from the nominal nonideal model using $R_{\mathrm{total}} = 4.0\ \Omega$, $V_{d,\mathrm{body}} = 0.7\ \mathrm{V}$, and the measured V_{in} and duty ratio. The voltage and current errors are

$$e_V = \frac{V_{\mathrm{out,measured}} - V_{\mathrm{out,predicted}}}{V_{\mathrm{out,predicted}}} \times 100\%, \tag{18}$$

$$e_I = \frac{i_{L,\mathrm{measured}} - i_{L,\mathrm{predicted}}}{i_{L,\mathrm{predicted}}} \times 100\%. \tag{19}$$

Table 5 shows the completed hardware verification table. The negative voltage errors indicate that the nominal loss assumptions are not sufficient for every operating point, especially at the lower input-voltage cases. This motivates the diagnosis step rather than invalidating the complete workflow.

Table 5: Predicted and measured converter behavior from the six uploaded datasets.

Test	V_{in}	D	$V_{\text{out,pred}}$	$V_{\text{out,meas}}$	e_V	$i_{L,\text{pred}}$	$i_{L,\text{meas}}$	e_I
10 V, $D = 0.55$	9.63	0.554	19.16	17.05	-11.0%	0.195	0.196	0.1%
10 V, $D = 0.60$	9.61	0.603	21.08	19.59	-7.1%	0.242	0.252	4.4%
12 V, $D = 0.55$	11.57	0.553	23.09	21.41	-7.3%	0.235	0.244	3.9%
12 V, $D = 0.60$	11.58	0.602	25.49	24.46	-4.0%	0.291	0.313	7.6%
15 V, $D = 0.55$	14.56	0.552	29.16	27.86	-4.5%	0.296	0.318	7.3%
15 V, $D = 0.60$	14.53	0.601	32.08	31.72	-1.1%	0.366	0.405	10.6%

3.4 PANN Diagnosis Verification

The PANN model is verified only in diagnosis mode. It is not used as the main forward-prediction engine because the forward boost-converter behavior is already described by the interpretable nonideal model. Instead, PANN is used when measurements differ from the prediction and hidden circuit parameters must be estimated from data. This follows the physics-in-architecture idea, where learnable quantities correspond to circuit parameters rather than arbitrary black-box weights [2].

The uploaded archive did not include a saved trained model output file, so Table 6 reports data-backed diagnosis estimates computed from the same physical equations used by the PANN structure. The inductance estimate is obtained from raw current-slope regression, the capacitance estimate is obtained from period-averaged capacitor-current regression, and the loss terms are obtained from steady-state regression across the six operating points.

If the measured output voltage is lower than the predicted value, the likely causes are a larger effective diode-related drop, duty-cycle mismatch, or an installed passive component that differs from the nominal value. If the measured inductor current is higher than expected, the load resistance, duty-cycle command, current-sense calibration, and installed inductance should be checked first.

3.5 Verification Summary

The completed data tables show that the software-to-hardware verification path is functional. PE-GPT can plan the boost-converter design task, call the nonideal physical model, send a duty-cycle command, compare measured and predicted converter behavior, and use measurement data to support diagnosis. The six uploaded datasets also show a consistent trend: the measured output voltage is below the nominal prediction at low input voltage and becomes closer to the prediction as V_{in} increases.

This verification structure matches the final project objective. The physical model is used where the boost-converter physics is known, and PANN is reserved for estimating hidden hardware parameters from experimental data. Therefore, the completed system is verified as both a working boost-converter design path and an LLM-centered circuit-design agent.

Table 6: Diagnosis estimates from the six uploaded datasets.

Parameter	**Nominal value**	**Diagnosis estimate**	**Diagnostic meaning**
L	720 μH calc.; 1 mH selected	648.5 μH	Effective inductance is closer to the calculated value than to the selected inventory module.
C	217 μF calc.; 220 μF selected	99.0 μF	Effective capacitance appears closer to a 100 μF installed value, which affects ripple and transient response.
R_{total}	4.00 Ω	0.87 Ω	Lower fitted series resistance suggests that voltage loss is not dominated by ohmic resistance alone.
$V_{d,\text{body}}$	0.70 V	4.55 V effective	The high effective drop explains why measured V_{out} is below the nominal prediction.
V_{SS34}	0.20 V assumed	0.20 V	The fitted input-path drop is consistent with a low-current Schottky drop.

4 Costs

4.1 Parts

Table 7 lists the updated prototype parts cost for the final hardware implementation. The final prototype used WeAct GD32F303CCT6 controller boards, CH340 serial modules, a laboratory-owned GaN half-bridge evaluation board, external sensing hardware, and interchangeable passive components. Laboratory-owned instruments and boards, including the GSP65R13HB-EVB half-bridge board, are listed with zero direct project cost. Receipt-based prices should be inserted before final submission for the purchased Taobao modules and passive components.

Table 7: Updated prototype parts costs.

Part	Quantity	Unit cost (RMB)	Actual cost (RMB)
WeAct GD32F303CCT6 GD32F development board	3	18.90	56.70
CH340 USB-to-serial module	2	4.79	9.58
GSP65R13HB-EVB GaN half-bridge evaluation board	1	Lab-owned	0
Voltage/current sampling module	1	92.00	92.00
Interchangeable inductors for converter testing	3	20.00	60.00
Interchangeable output capacitors for converter testing	3	8.00	24.00
Power load resistors for converter testing	3	15.00	45.00
PCB manufacturing and assembled components	1 set	200.00	200.00
Total			487.28

4.2 Labor

The labor cost is estimated separately from the direct hardware cost because most project effort was spent on design, software integration, hardware debugging, data collection, and report preparation. Following the ECE 445 cost-estimation method, each member's labor cost is

$$C_{\mathrm{labor},i} = S_i H_i \times 2.5, \tag{20}$$

where S_i is the ideal hourly salary, H_i is the project time, and the 2.5 multiplier accounts for overhead, benefits, and engineering support. Table 8 should be completed with the final logged hours and selected hourly rate before submission.

Table 8: Labor cost estimate for the final project.

Team member	Primary project contribution	Ideal salary (USD/h)	Hours	Labor cost (USD)
Jiarong Xu	PE-GPT workflow, system integration, report organization	Insert rate	Insert hours	Insert cost
Haojun Li	Nonideal boost model, sizing equations, design-tool verification	Insert rate	Insert hours	Insert cost
Xinyu Zhan	GD32/CH340 communication, PWM path, hardware data acquisition	Insert rate	Insert hours	Insert cost
Jiaming Ma	PANN training, diagnosis workflow, data processing, verification	Insert rate	Insert hours	Insert cost
Total	–	–	Insert total hours	Insert total labor cost

For example, if $S_i = \$40/\mathrm{h}$ and $H_i = 120\ \mathrm{h}$, then

$$C_{\mathrm{labor},i} = 40 \times 120 \times 2.5 = \$12{,}000. \tag{21}$$

The total project cost is the sum of the direct prototype parts cost in Table 7 and the total labor cost in Table 8. This separation shows that the hardware prototype is relatively inexpensive, while the engineering effort for PE-GPT planning, PANN diagnosis, GD32 control, serial communication, and converter measurements is the dominant cost.

5 Conclusion

5.1 Accomplishments

This project developed a PE-GPT-based workflow assistant for intelligent power-electronics design. The most important accomplishment is the transformation of PE-GPT from a static numerical calculator into a staged engineering workflow agent.

The completed system demonstrates AI-guided topology understanding, workflow planning, structured and natural-language interaction, requirement validation, physics-based design, hardware-oriented reasoning, and diagnosis preparation.

The asynchronous boost-converter workflow is currently the most complete implementation. The system includes topology reasoning, nonideal physical modeling, staged interaction, hardware preparation, and diagnosis activation. Unlike ideal boost-converter demonstrations, the implemented workflow explicitly considers nonideal quantities such as V_{ss34}, $V_{d,body}$, and R_{total}.

Another important accomplishment is the integration of planning-oriented reasoning into the interaction process. Instead of directly exposing all workflow stages at the beginning, the assistant dynamically generates and progressively expands the design procedure according to the selected task.

5.2 Current Limitations

Several limitations remain in the current implementation.

First, only the asynchronous boost-converter branch is fully implemented. Although buck converters and RLC oscillators already share the same planning framework, their detailed physical models and diagnosis procedures are not yet completed.

Second, the hardware-validation stage still depends partially on manual interaction. Some measurements and confirmations must currently be provided by the user instead of being acquired automatically from the hardware interface.

Third, the diagnosis module remains a prototype. More measured datasets are required before the PANN-based estimation results can be considered fully validated.

Finally, the current system is still a guided assistant rather than a fully autonomous design platform. Human confirmation remains necessary at critical hardware stages for both safety and reliability reasons.

5.3 Future Work

Future work should focus on tighter integration between AI planning, hardware interaction, and online diagnosis.

The first direction is automatic hardware communication. Future versions should directly configure controller parameters and acquire waveform measurements from the hardware

platform automatically.

The second direction is improved diagnosis capability. With more measured data, the PANN-based module can estimate hidden parasitic quantities more accurately and provide stronger explanations for prediction-measurement mismatch.

The third direction is multi-topology expansion. The same workflow structure can be extended to buck converters, resonant converters, and more general RLC networks by attaching different physical models and validation procedures.

Another important direction is multimodal reasoning. Future versions may allow the assistant to analyze oscilloscope screenshots, waveform plots, PCB images, and experimental videos in addition to numerical data files.

Overall, this project demonstrates that combining LLM planning, physics-based tools, staged reasoning, and hardware-oriented interaction is a promising direction for intelligent power-electronics design assistance.

References

[1] R. W. Erickson and D. Maksimović, *Fundamentals of Power Electronics*, 2nd ed. New York: Kluwer Academic Publishers, 2001.

[2] X. Li et al., "Temporal Modeling for Power Converters with Physics-in-Architecture Recurrent Neural Network," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 11, pp. 14 111–14 123, 2024. DOI: 10.1109/TIE.2024.3352119