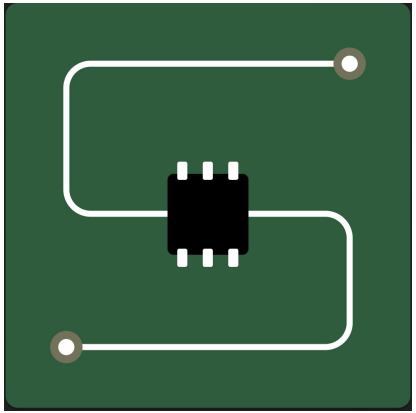
Simantic
December 2025



Test Hardware Without Hardware

Introducing Simantic:

We are building software for hardware engineers.

Software to test and validate hardware, without the hardware.

All in simulation.

Meet The Founder

Seungmin Hong
Founder

Background:

University of Waterloo, Computer Engineering

Simbe Robotics
Firmware Engineer

Developed battery, IMU firmware for prototype robots

Tesla
Firmware Engineer

New Programs Engineering
Prototype vehicles and cool features

Keys and Authentication
OTA update protocol design for robotaxis

Rapyuta Robotics
Firmware Engineer

NVIDIA Jetson integration and simulation

Ericsson
ML Software Developer

ML model development and training for early failure detection

Challenges in Hardware Engineering

01

Hardware Design is Expensive

Designing and testing hardware is expensive. Mistakes in hardware design can cost thousands in material and labour costs.

02

Hardware Design is Lagging

Unlike the rise of AI tools in the software industry, innovation in hardware engineering is lacking in comparison.

03

Hardware Design is Inaccessible

Due to the cost associated with hardware design, individuals and small teams have difficulty breaking into the industry.

WHAT SIMANTIC DOES

01

Run Firmware on Virtual Processors

Ensure that your board does exactly what you want it to, by running code on a virtual model of the processor, before it even reaches manufacturing.

02

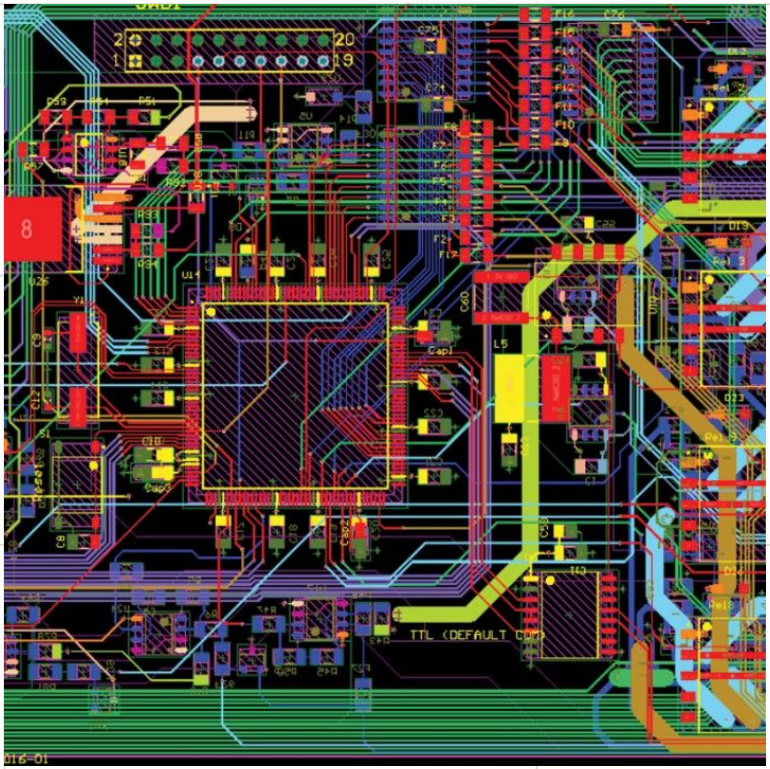
Fully Simulate the PCB Behavior

Simulate the electrical signal behavior on your board as it interacts with the virtual processor. Physics and mathematics based models to ensure that signal integrity and power needs are met.

03

AI Assisted Design and Code Generation

Use AI to help design your PCB, and generate code that goes hand-in-hand with each other.



Product Intro: CI/CD for Hardware + Firmware

01

Dynamic Firmware Runtime Analysis

Test firmware in virtual hardware without the hardware.

Detect errors that static analysis misses, such as multithreading, memory, and runtime performance.

02

Supporting CPU Models and Architectures

Support for most modern instruction sets (ARM, RISC-V, x86, Xtensa, etc.).

Growing list of supported CPU models and peripheral support.

Deterministic code execution in simulation, matches physical hardware behavior.

03

Easy Development Workflow Integration

Testing code is as simple as pushing to a Github repository.

View test results and performance benchmarks in a web UI, with detailed explanations for detected issues, through CPU trace analysis.

```c
void StartUartTask(void *argument)
{
  /* USER CODE BEGIN StartUartTask */
  const char *message = "Hello from FreeRTOS task! Uptime: ";
  const char *units = " seconds\r\n";
  char buffer[64];
  uint32_t counter = 0;

  /* Infinite loop */
  for(;;)
  {
    /* Prepare the message with counter */
    strcpy(buffer, message);

    /* Convert counter to string and append */
    char counterStr[16];
    sprintf(counterStr, "%lu", counter);
    strcat(buffer, counterStr);
    strcat(buffer, units);

    /* Send the message over UART */
    HAL_UART_Transmit(&huart1, (uint8_t*)buffer, strlen(buffer), 1000);
```

```
08001064 <StartUartTask>:
 8001064:   b580        push    {r7, lr}
 8001066:   b09a        sub sp, #104   @ 0x68
 8001068:   af00        add r7, sp, #0
 800106a:   6078        str r0, [r7, #4]
 800106c:   4b1b        ldr r3, [pc, #108] @ (80010dc <StartUartTask+0x78>)
 800106e:   663b        str r3, [r7, #96]   @ 0x60
 8001070:   4b1b        ldr r3, [pc, #108] @ (80010e0 <StartUartTask+0x7c>)
 8001072:   65fb        str r3, [r7, #92]   @ 0x5c
 8001074:   2300        movs    r3, #0
 8001076:   667b        str r3, [r7, #100] @ 0x64
 8001078:   f107 031c   add.w   r3, r7, #28
 800107c:   6e39        ldr r1, [r7, #96]   @ 0x60
 800107e:   4618        mov r0, r3
 8001080:   f004 fb08   bl  8005694 <strcpy>
 8001084:   f107 030c   add.w   r3, r7, #12
 8001088:   6e7a        ldr r2, [r7, #100] @ 0x64
```

Product Intro: PCB Software Development Kit

01

Interact with your PCB Design in Code

We are developing a compiler for your PCB project files.

Whether you design with Altium, KiCad, or another CAD tool, our PCB compiler will turn your design into software libraries, which can be imported into modern programming languages.

```python
def test_overvoltage_transience(board: PCBObject):
    board.reset()
    board.input1.set_voltage(10, "impulse")
    board.c1.measure_current(t = 100, interval = 1)
    board.h1.measure_current(t = 100, interval = 1)
```

02

Unit Tests for Hardware

Perform advanced industry-standard tests on your PCB designs in software, without waiting for a physical prototype.

Tests like power/signal integrity tests, transient analysis, and current measurements all require highly specialized and expensive tool. With Simantic, it's just a function call away.

```python
def setup_machine(machine_name):
    machine1 = emu.add_mach(machine_name)
    machine1.load_repl(str(Path("example.repl").resolve()))
    machine1.load_elf(str("example.elf"))

    all_uarts = get_all_uarts(machine1)
    if len(all_uarts) > 0:
        for uart, uart_name in all_uarts:
            uart.CharReceived += create_callback(uart, f"{machine_name}_{uart_name}")
```

03

Firmware Co-Simulation

Test your PCB by running the firmware simulation to test interactions between the CPU and board.

Reduce design iteration and validation downtown by not having to wait for a finalized hardware revision to test firmware.

https://simantic.dev

Simantic = Simulation + Semantics

Semantics Definition:

Semantics in the context of programming language theory and artificial intelligence (AI) involves the formal study of the meanings of programming constructs, beyond their syntactic form. It bridges the gap between the abstract, textual representation of programs (syntax) and their operational behaviors on a computing platform (semantics).

Thank you for listening. More to come soon.

CONTACT
seungmin@simantic.dev
https://www.linkedin.com/company/simantic