Gary Gao

——————— | g44gao@uwaterloo.ca | in/garygaoce | GaryGaoUW | ggao.dev

Technical Skills

Languages: C/C++, Python, Verilog, C++/CLI (.NET)
Microcontrollers & Arch: STM32, ESP32 (ESP-IDF), ARM Cortex, RISC-V, FPGA (Artix-7)
Protocols & Frameworks: FreeRTOS, UART, SPI, I2C, Modbus, BLE, MQTT, AXI4-Lite, OpenCV, LVGL
Tools & Instruments: Oscilloscope, Logic Analyzer, JTAG, GDB, Git, Linux, Altium Designer, Xilinx Vivado, Verilator, Cocotb

Education

University of Waterloo Sep 2024 – Present
Applied Science Computer Engineering, Honours, Co-operative Program | Cumulative Average: 82% *Waterloo, Ontario*

Experience

Embedded Engineer – Test Fixture | *C++, C++/CLI (.NET), Python, OpenCV* Sep 2026 – Dec 2026
Ecobee - Onsite *Toronto, Ontario*

- Root-caused LCD test false fails to camera saturation; modeled camera gamma ($\gamma \approx 0.45$) from an exposure sweep and derived a 16 ms exposure limit (from 30 ms), cutting clipped pixels from **75% to 0%**.
- Replaced pixel-wise **MSE** with **Rec.709** luma, **R/G, B/G**, and per-bar color checks, prototyped in **Python** and deployed in **C++ OpenCV**; validated by fault injection (schematic-guided current-regulator and resistor rework), catching tinted and broken screens.
- Built **Regex**-based device-descriptor parsers and barcode builders in the shared **C++/CLI** core, auto-binding COM ports and supporting **6 SKU/RMA** variants without hard-coded station configs.
- Ran a crossed **Gage R&R** (6 nests $\times$ 10 PCBAs $\times$ 3 = **180 runs**); **ANOVA** isolated a fixed nest-to-nest offset within tolerance, clearing fixtures for factory release.

Embedded Systems Engineer | *C/C++, FreeRTOS, Modbus, Altium, ESP32, STM32* Jan 2026 – Apr 2026
Food Cycle Science - Onsite *Ottawa, Ontario*

- Migrated an Arduino polling codebase to event-driven **ESP-IDF** on **ESP32** with a non-blocking **FSM**; preserved **AWS IoT MQTT** for OTA and remote control, and **BLE** (NimBLE) provisioning of device keys and Wi-Fi credentials.
- Architected a **FreeRTOS** system (preemptive scheduler, thread priorities, stack sizing) using **IPC** mutexes and queues for thread-safe shared memory, preventing race conditions.
- Engineered a custom **STM32** bootloader and fault-tolerant **OTA** designed to **EU TÜV** requirements: signature authentication, **SHA-256** integrity checks, dual-partition fallback, and power-loss resume.
- Implemented **UART** and **Modbus** with custom frame parsing; debugged register-level hex streams, timing, and multi-node states via oscilloscope, logic analyzer, and **JTAG**.
- Designed board schematics in **Altium Designer** with high-frequency impedance matching, transient voltage protection, and high/low-voltage routing isolation.
- Built a 23-screen **LVGL** GUI; prevented **SRAM** heap fragmentation via strict lifecycle management and delete-before-create rendering.

Projects

RISC-V SoC with AXI4-Lite Bus | *Verilog, Python, AXI4-Lite, Verilator, Cocotb* Sep 2026 – Present

- Built a **Verilator** C++ harness and **Python** regression for **riscv-tests** (`ecall`/`a0` checks, hang watchdog); verified single-cycle, then 5-stage pipeline with forwarding and load-use stalls, both **38/38 rv32ui**.
- Hand-wrote a 5-state **AXI4-Lite** data master (AR/R/AW/W/B, `WSTRB`) and a read-only instruction master, turning core memory requests into valid/ready handshakes.
- Designed a 3-state AXI4-Lite RAM slave (write-priority arbitration, byte strobes), verified with **Cocotb**/**cocotbext-axi**.
- Integrated a memory-agnostic `soc_top` with a global stall for variable AXI latency; a latency-injecting bus scoreboard caught duplicate load/store requests missed by riscv-tests, fixed by a one-shot request gate (**0 extra transactions**).

RISC-V 5-Stage Pipelined Processor | *Verilog, Vivado, Artix-7* Mar 2026 – May 2026

- Synthesized a 32-bit **RV32I** core in **Verilog**, evolving single-cycle (18 ns, 55.6 MHz) into a 5-stage pipeline (**IF/ID/EX/MEM/WB**) closing timing at **15 ns (66.7 MHz), a 20% faster clock**.
- Engineered EX/MEM and MEM/WB forwarding with a Load-Store bypass that skips unnecessary Load-Use stalls, plus flush/stall hazard detection for branches, improving **IPC by 20%+** versus stalling on every data hazard.
- Migrated Distributed to Block RAM (2,031 $\rightarrow$ 1,884 LUTs) and broke a 16-level logic chain, achieving **100% timing closure** with a 14.81 ns critical path (WNS = +0.186 ns).

Bare-Metal LCD Driver & Breakout PCB | *C++, SPI, ESP8266, Altium* Oct 2024 – Mar 2025

- Wrote a **C++** GC9203 driver from the datasheet on a custom **Altium** PCB; debugged **SPI** bring-up with a **logic analyzer**.
- Halved RAM (56.3 $\rightarrow$ **28.7 KB**) via a 1-byte indexed framebuffer, RGB565 LUT, and dirty-row bitmap for partial refresh.