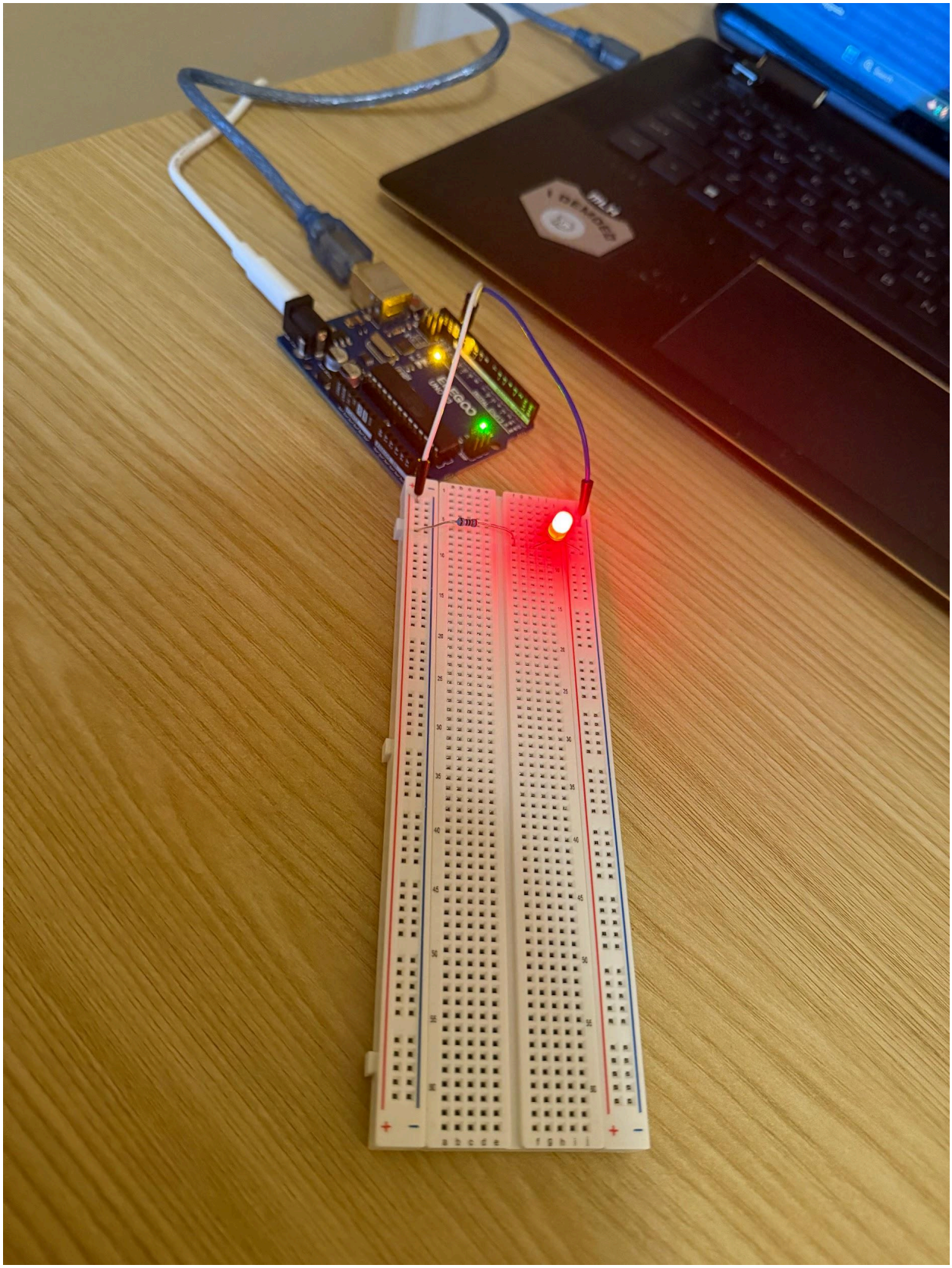
Day - 1 LED blinking

This project was conceptually straightforward and served as a good introduction to basic GPIO and timing control. Most of the time was spent setting up the development environment and understanding how the Arduino IDE communicates with the Arduino Uno. I initially faced difficulties uploading the code, as I believed the board was not connected properly, even though it was connected throughout the process. By troubleshooting the issue, I learned how to correctly select the board and port, upload code, and identify IDE-related errors. Once resolved, the LED blinking functionality worked as expected. Overall, this was a solid starting project that helped build familiarity with the tools and workflow, and it was completed ahead of schedule.

Skills Learned

- Configured and troubleshot the Arduino Uno development environment and IDE setup
- Gained hands-on experience with GPIO configuration using `pinMode()` and `digitalWrite()`
- Implemented timing control using software delays to generate periodic signals
- Understood the relationship between delay values, period, and output frequency
- Learned the process of compiling and uploading embedded code to a microcontroller
- Developed basic debugging skills by identifying connection and configuration issues

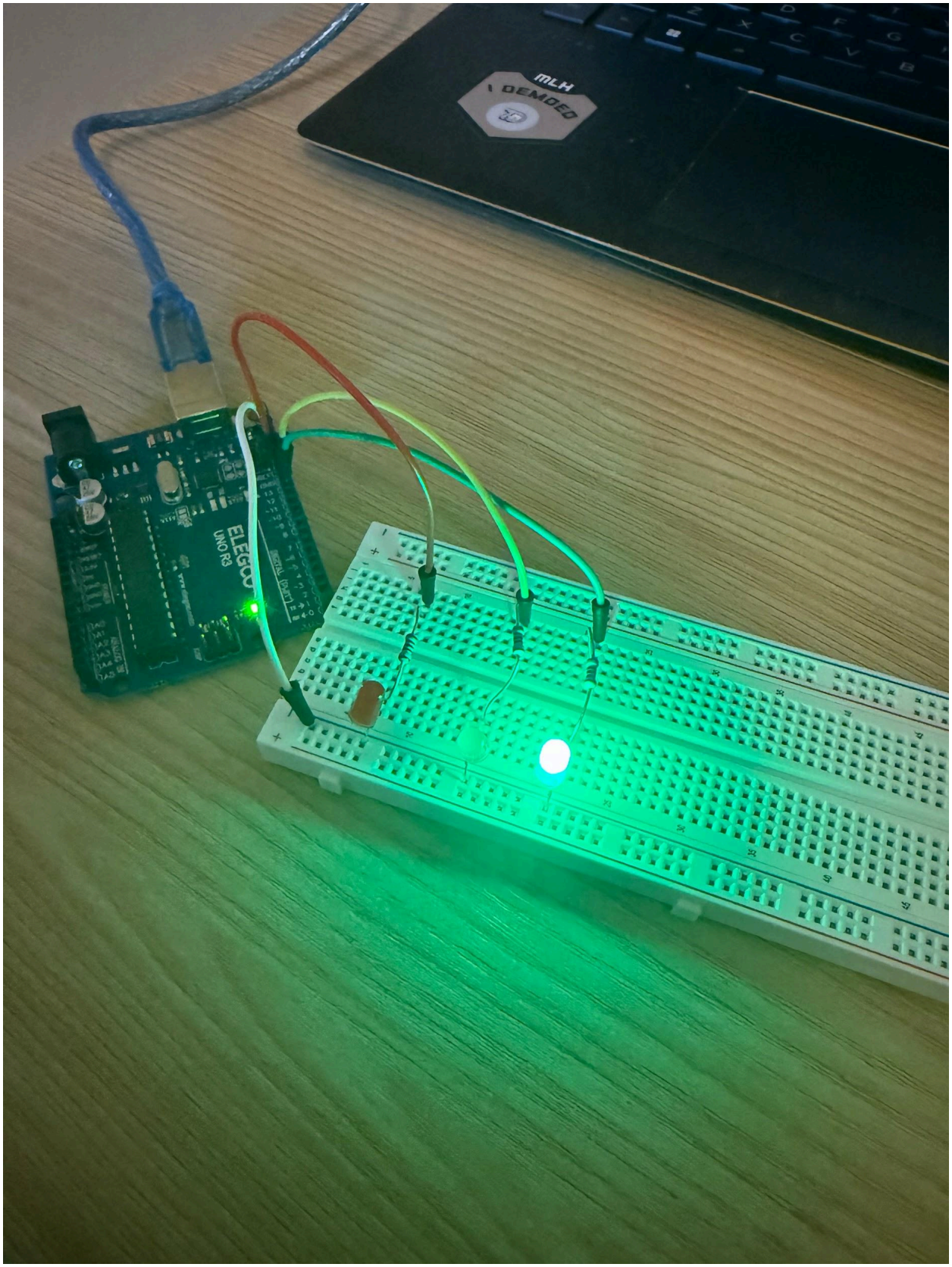


Day - 2: Traffic Lights

This project was relatively straightforward, as familiarity with the Arduino IDE and hardware setup significantly reduced setup time compared to the previous project. Implementing the traffic light controller was intuitive and reinforced the concept of structured control logic using multiple outputs. The project did not require extensive troubleshooting, allowing more focus on understanding the behavior of the system and verifying correct state transitions. Overall, the project was enjoyable to implement and helped strengthen confidence in designing simple state-based control systems.

Skills Learned

- Designed and implemented a simple finite state machine (FSM) using sequential logic
- Controlled multiple GPIO outputs simultaneously for coordinated system behavior
- Applied structured timing logic to manage state transitions
- Strengthened understanding of embedded system control flow
- Improved code organization for readability and scalability
- Reinforced debugging and verification of expected hardware behavior

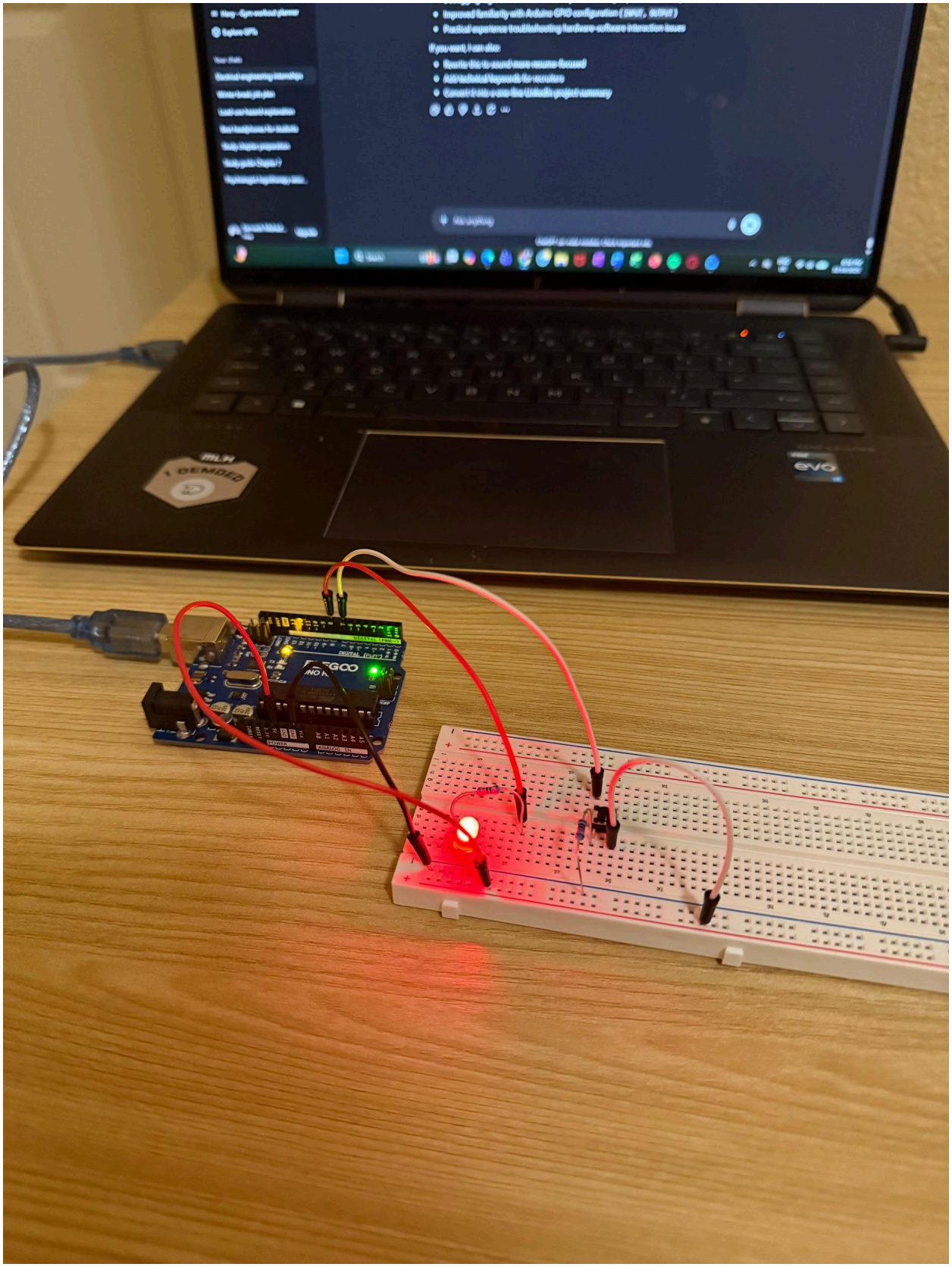

MLH
I DEMOED
UNO R3

Day - 3: Toggle and debounce with a button and LED -

This project was more challenging than the previous ones, as it required a deeper understanding of digital inputs and software debouncing. I spent additional time reviewing the material to fully grasp the concepts involved. During development, I encountered several issues where the LED behavior was inconsistent—at times it would not turn on, and at other times it would not turn off as expected. Through manual debugging and careful testing of the logic, I was able to identify and resolve these issues. The initial hardware and software setup also took longer than anticipated, primarily due to user error, but it ultimately helped reinforce my understanding of the development workflow. Overall, this project was a valuable learning experience and strengthened my debugging and problem-solving skills.

Skills Learned

- Understanding and implementation of digital input using push buttons
- Software debouncing techniques to handle noisy button signals
- Use of `millis()` for non-blocking timing and state tracking
- Debugging logic errors in embedded systems through systematic testing
- Improved familiarity with Arduino GPIO configuration (`INPUT`, `OUTPUT`)
- Practical experience troubleshooting hardware–software interaction issues



DAY 4 — Software Debouncing Logic -

This project focused on improving button input reliability by implementing a timing-based software debouncing algorithm. Unlike earlier approaches, the logic separates raw input changes from confirmed button states, ensuring state transitions are only registered after remaining stable for a defined debounce interval. This significantly improved system reliability and eliminated inconsistent LED behavior caused by mechanical button noise. The project reinforced the importance of robust input handling in embedded systems and introduced scalable design practices applicable to more complex digital interfaces.

Skills Learned

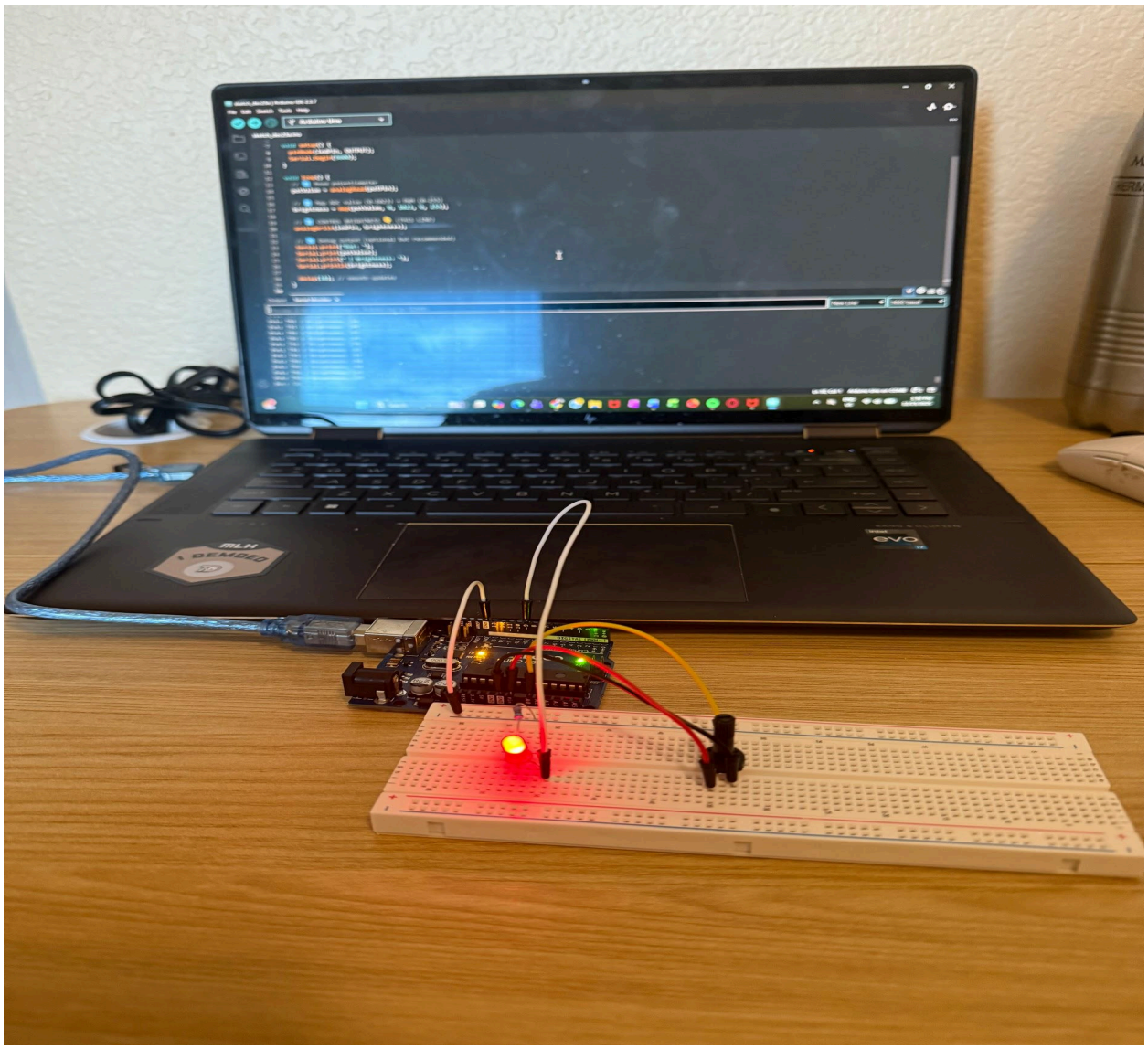
- Advanced software debouncing techniques
- Use of `millis()` for precise, non-blocking timing
- Designing reliable state-change detection logic
- Writing scalable and reusable embedded code
- Understanding real-world hardware signal noise
- Improving system robustness through software design

DAY 5 — Analog Sensor Read (Potentiometer)

This project introduced analog-to-digital conversion by reading a potentiometer input and mapping the resulting digital values to corresponding voltage levels. Using the Arduino's 10-bit ADC, analog voltages between 0 and 5V were converted into discrete digital values, which were then scaled and visualized through serial output and optional PWM-based LED control. This project provided foundational insight into sensor interfacing and mixed-signal system behavior.

Skills Learned

- Analog-to-digital conversion fundamentals
- Voltage scaling and quantization
- Reading and processing analog sensor data
- Serial data visualization and debugging
- Mapping analog inputs to digital outputs

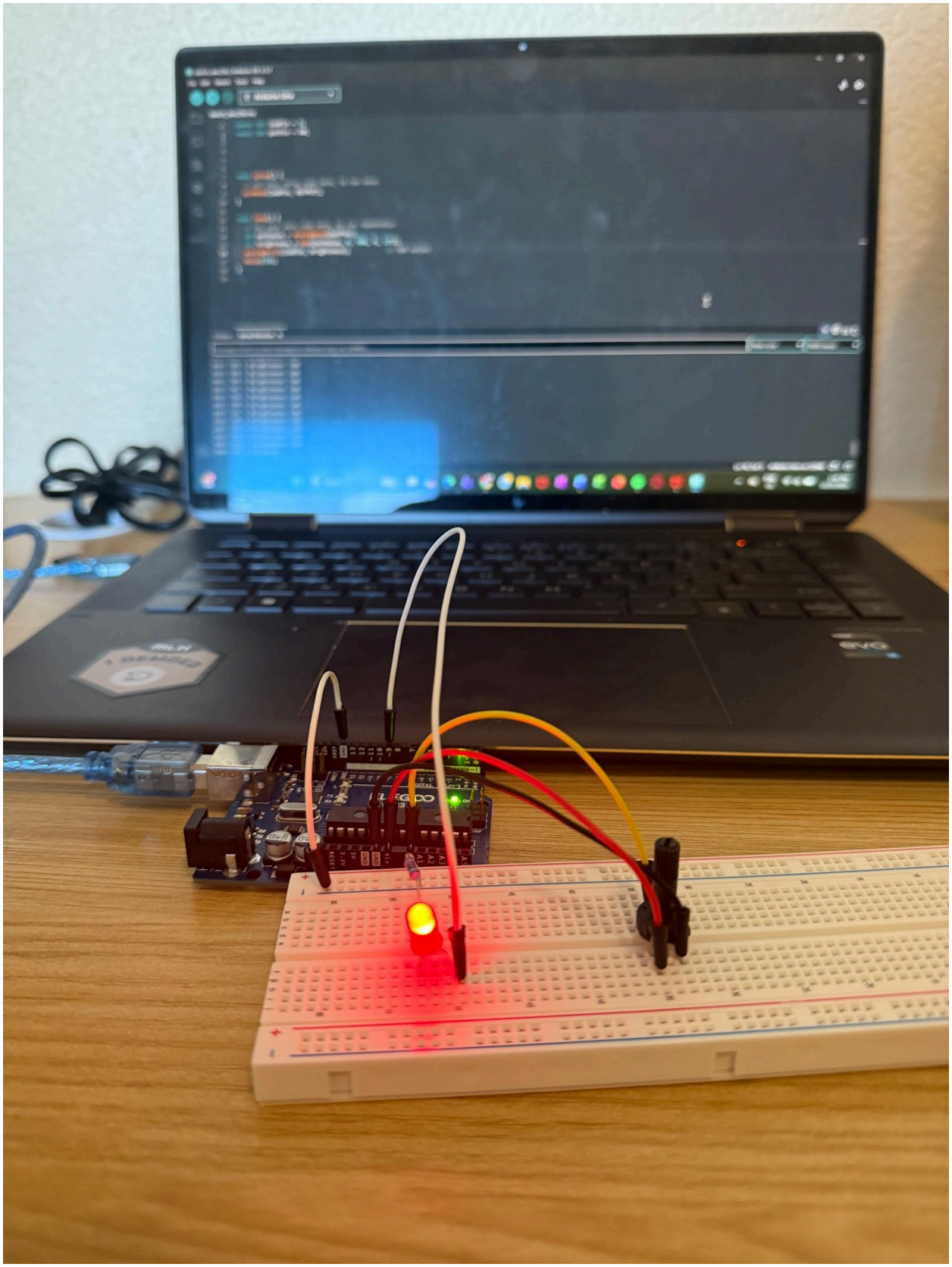


DAY 6 — PWM Brightness Control (LED Dimmer)

This project focused on implementing PWM-based LED dimming to understand how digital systems approximate analog behavior. By mapping ADC input values to an 8-bit PWM output, the LED brightness was smoothly controlled. This project reinforced core concepts of duty cycle modulation and demonstrated how PWM is used in embedded systems to regulate power and signal intensity.

Skills Learned (Bullet Points)

- Pulse Width Modulation (PWM) fundamentals
- Duty cycle control and brightness scaling
- ADC to PWM signal mapping
- Embedded systems signal control
- Practical microcontroller timing behavior

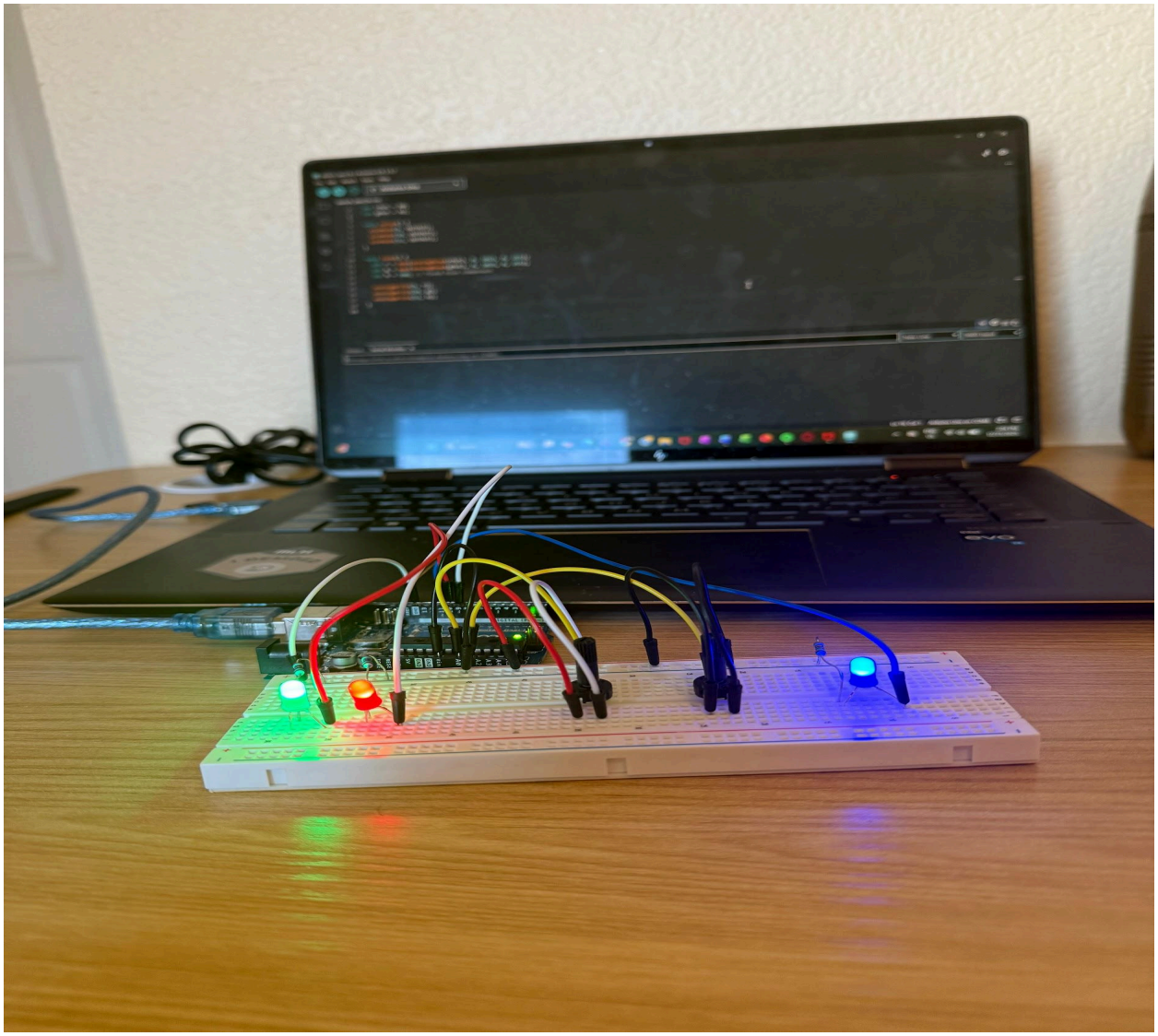


Day 7 — RGB LED Color Mixer

In this project, I implemented an RGB LED color mixer using PWM signals generated by the Arduino. Due to limited analog input resources, two potentiometers were used to control the red and green color channels, while the blue channel was held at a fixed intensity. This setup still enabled a wide range of visible color combinations and demonstrated how duty-cycle modulation affects perceived color mixing. The project reinforced practical constraints commonly encountered in embedded systems and highlighted the importance of making design trade-offs when hardware resources are limited.

Skills Learned

- PWM-based brightness and color control
- RGB color mixing and duty-cycle modulation
- Analog-to-digital conversion (ADC) usage
- Resource-aware embedded system design
- Mapping sensor inputs to output signals
- Debugging and tuning real-time embedded behavior

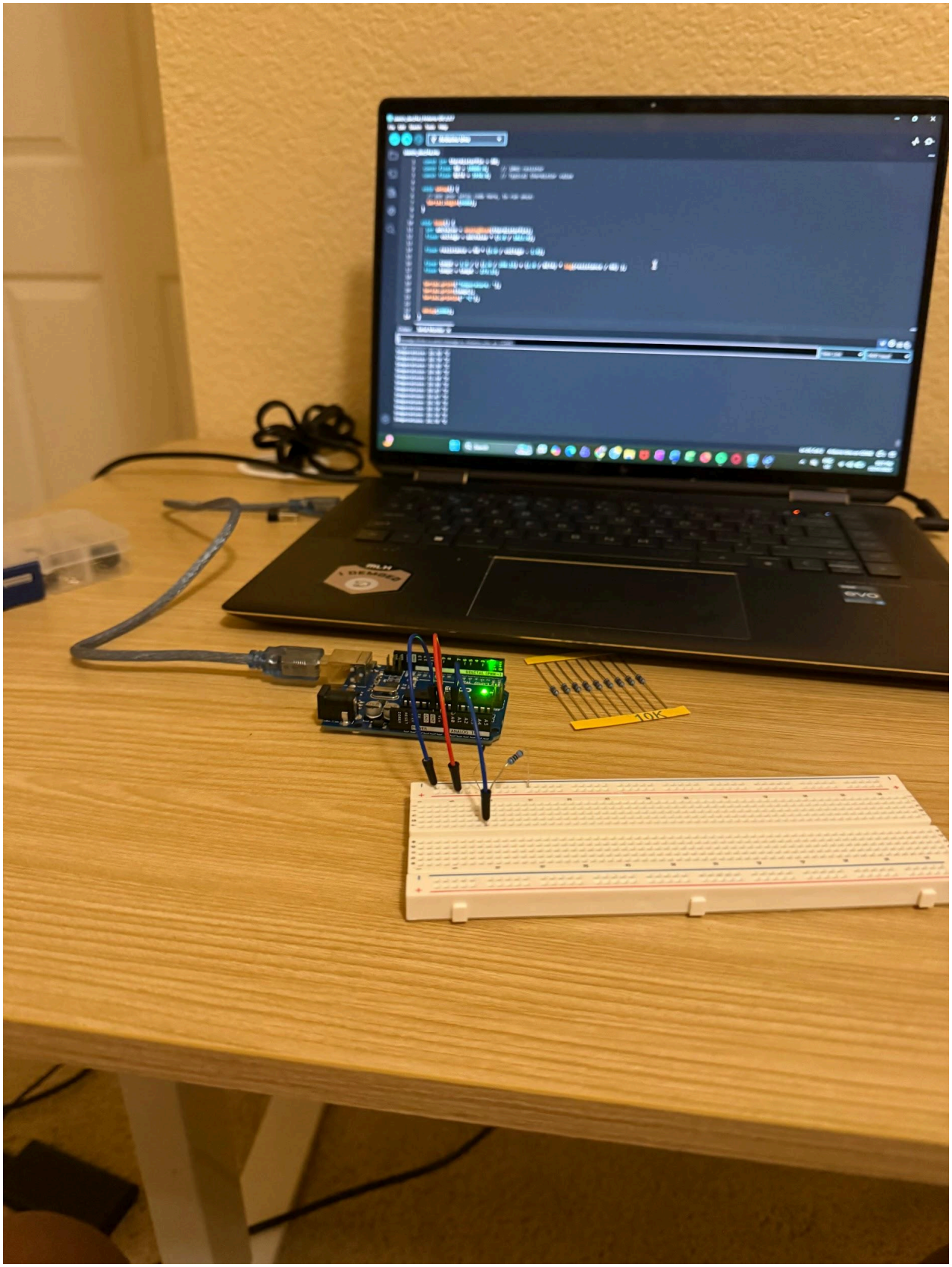


DAY 8 — Temperature Sensor Display (Thermistor)

This project introduced temperature sensing using a thermistor and demonstrated how analog sensor data can be converted into meaningful physical measurements through mathematical modeling. By implementing a voltage divider and applying logarithmic temperature calculations, I was able to compute and display real-time temperature values via the serial monitor. This project strengthened my understanding of sensor modeling, ADC behavior, and real-world data interpretation in embedded systems.

Skills Learned

- Thermistor-based temperature sensing
- Voltage divider circuits
- Analog-to-digital conversion (ADC)
- Mathematical modeling of physical sensors
- Serial data monitoring and debugging
- Embedded data acquisition principles

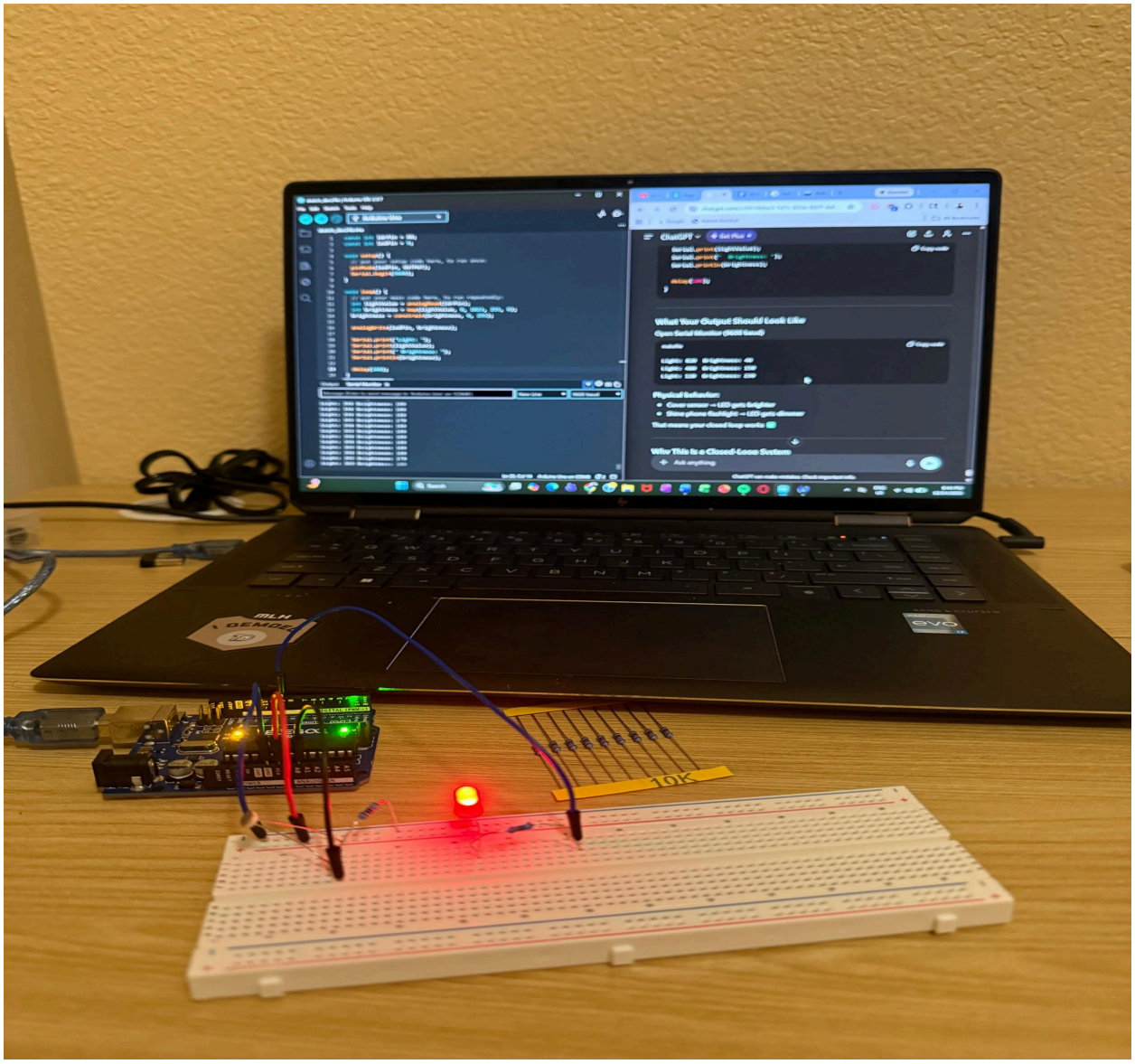


DAY 9 — Light Sensor + LED Auto-Brightness

This project implemented a closed-loop control system using a light sensor and PWM-controlled LED to automatically adjust brightness based on ambient lighting conditions. By continuously sensing environmental light levels and mapping the input to an output duty cycle, the system demonstrated real-time feedback control. This project strengthened my understanding of sensor-actuator feedback loops, control logic, and practical PWM-based signal modulation in embedded systems.

Skills Learned

- Closed-loop control system fundamentals
- Sensor-actuator feedback design
- PWM signal control using `analogWrite()`
- ADC to DAC concept mapping
- Real-time embedded system behavior
- Practical control theory implementation

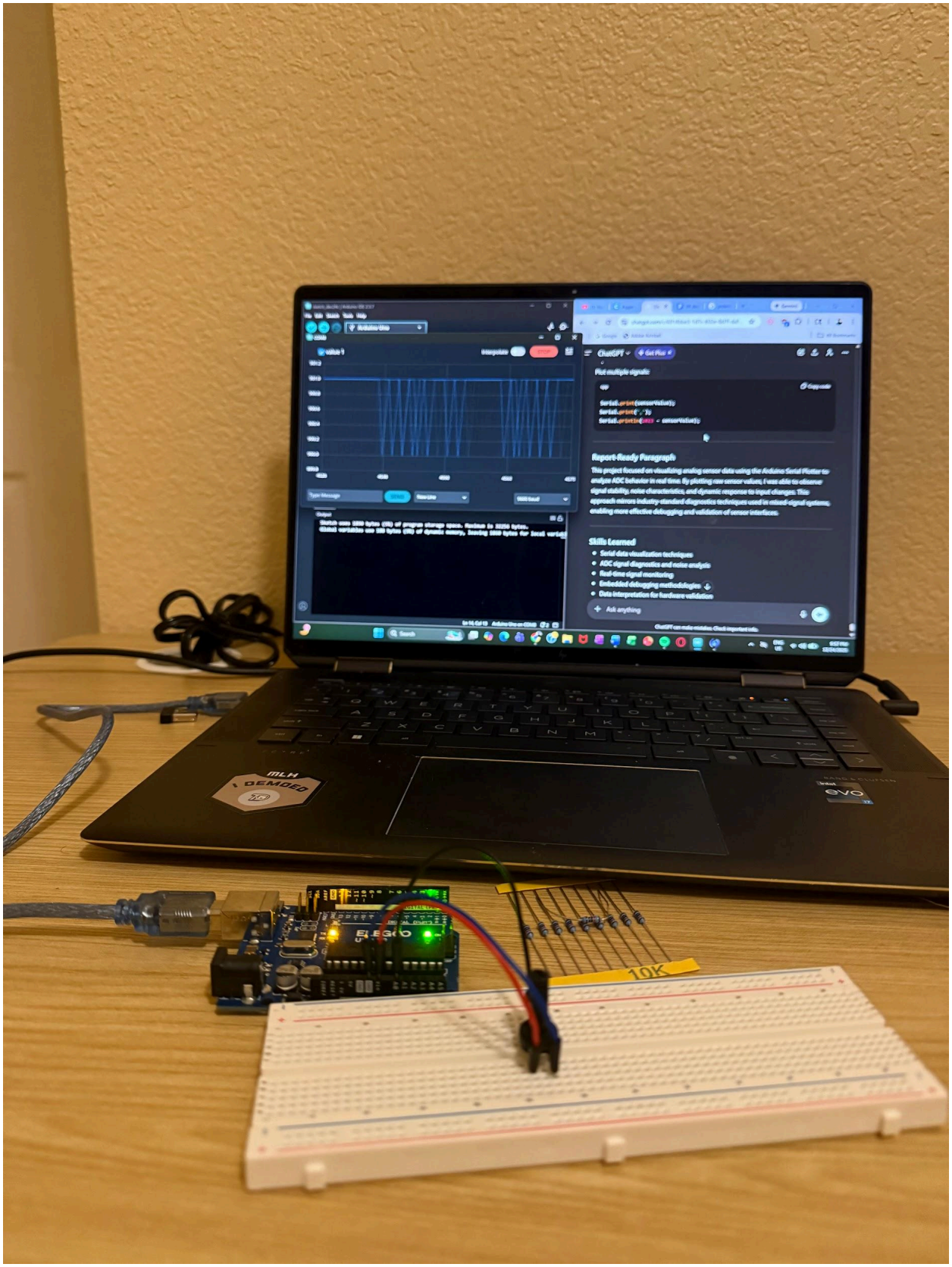


DAY 10 — Serial Plotting of Sensor Data

This project focused on visualizing analog sensor data using the Arduino Serial Plotter to analyze ADC behavior in real time. By plotting raw sensor values, I was able to observe signal stability, noise characteristics, and dynamic response to input changes. This approach mirrors industry-standard diagnostics techniques used in mixed-signal systems, enabling more effective debugging and validation of sensor interfaces.

Skills Learned

- Serial data visualization techniques
- ADC signal diagnostics and noise analysis
- Real-time signal monitoring
- Embedded debugging methodologies
- Data interpretation for hardware validation

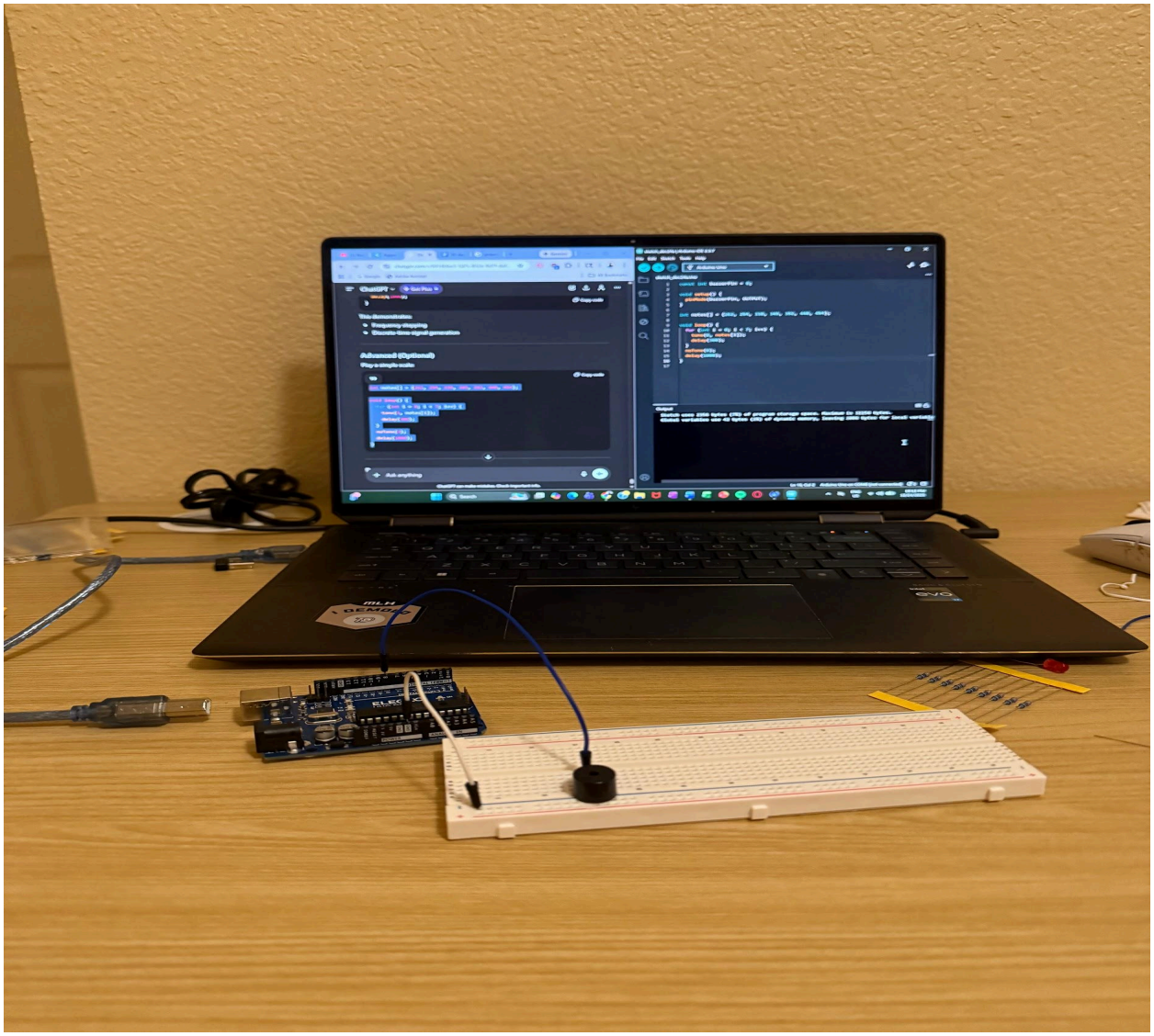

Plot multiple signals
Report-Ready Paragraph
Skills Learned
Ask anything
MLH
DEMOED
10K

DAY 11 — Buzzer Sound Generator

This project explored digital frequency generation using an Arduino-controlled buzzer. By programming different tone frequencies, I learned how digital systems create periodic signals that translate into physical outputs. The project reinforced concepts of timing, frequency control, and signal generation, which are foundational to clock design, PWM systems, and audio signal processing in embedded and digital hardware systems.

Skills Learned

- Frequency generation using digital timers
- Understanding tone-to-frequency mapping
- Timing control and delay-based scheduling
- Fundamentals of digital signal generation
- Hardware-software interaction in embedded systems



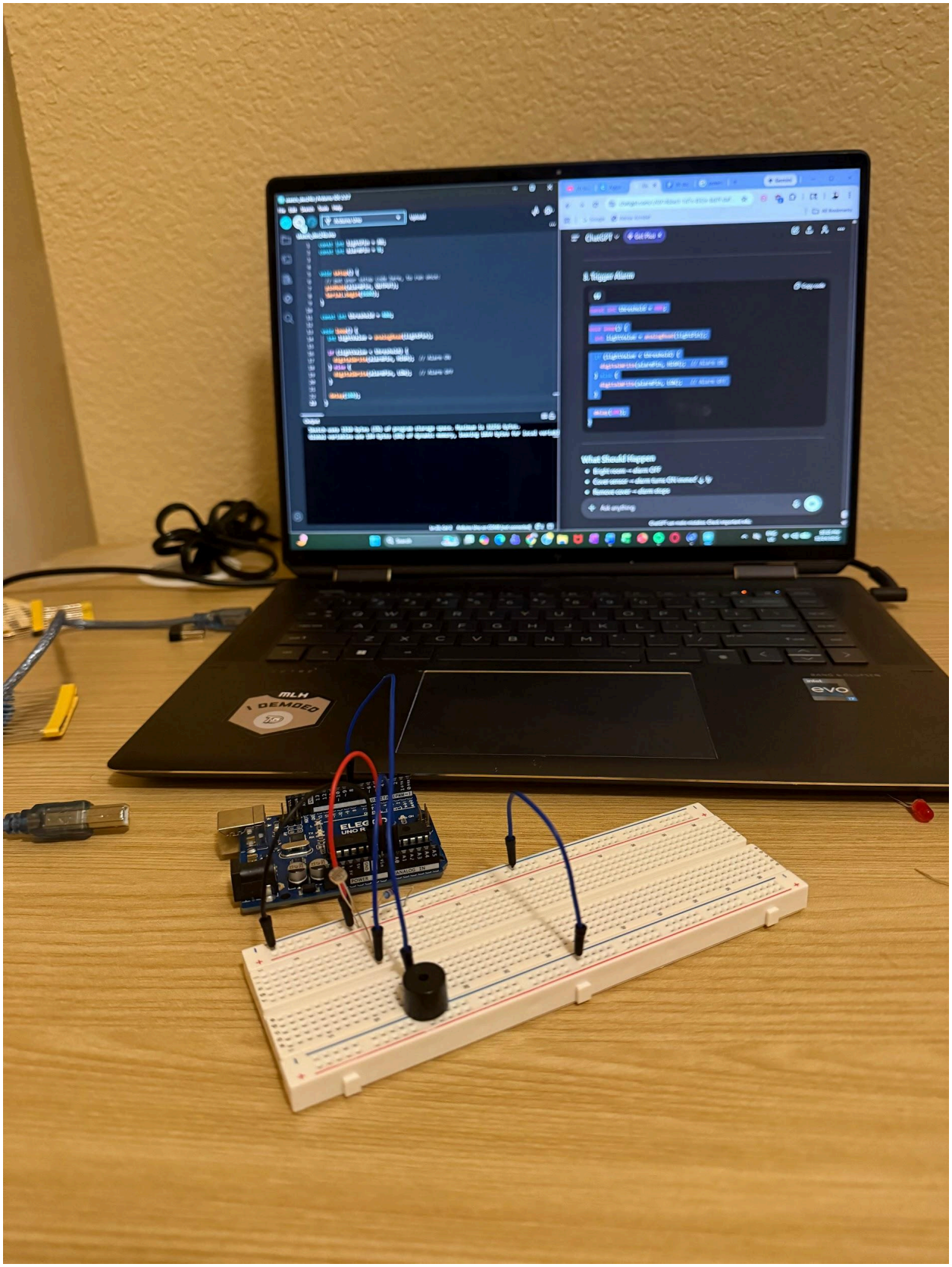
DAY 12 — Alarm Triggered by Light Change

This project implemented a light-based alarm system using threshold detection on analog sensor data. By continuously monitoring ambient light levels and comparing them against a defined threshold, the system triggered an alarm when a significant change was detected. This project reinforced the concept of signal conditioning and decision logic, which closely parallels the operation of analog comparators and event-driven systems in digital hardware.

Skills Learned

- Analog signal acquisition and conditioning

- Threshold-based decision logic
- Closed-loop event detection
- Debugging sensor-driven systems
- Mapping physical signals to digital control

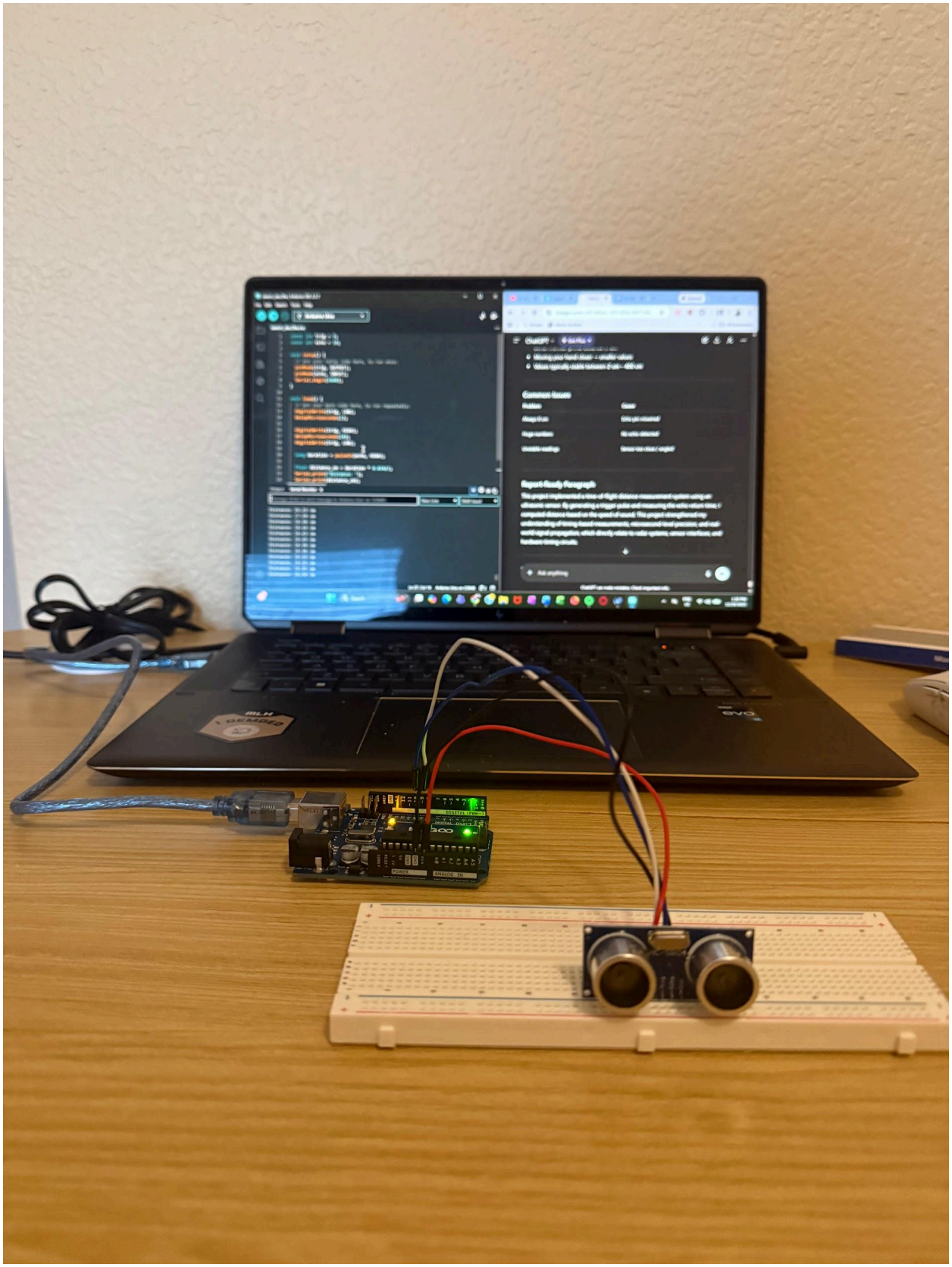

MLH
DEMOED

DAY 13 — Ultrasonic Distance Sensor

This project implemented a time-of-flight distance measurement system using an ultrasonic sensor. By generating a trigger pulse and measuring the echo return time, I computed distance based on the speed of sound. This project strengthened my understanding of timing-based measurements, microsecond-level precision, and real-world signal propagation, which directly relate to radar systems, sensor interfaces, and hardware timing circuits.

Skills Learned

- Time-of-flight measurement techniques
- Microsecond-level timing and pulse measurement
- Timer-based computation and signal latency analysis
- Sensor interfacing and calibration
- Translating physical phenomena into digital data

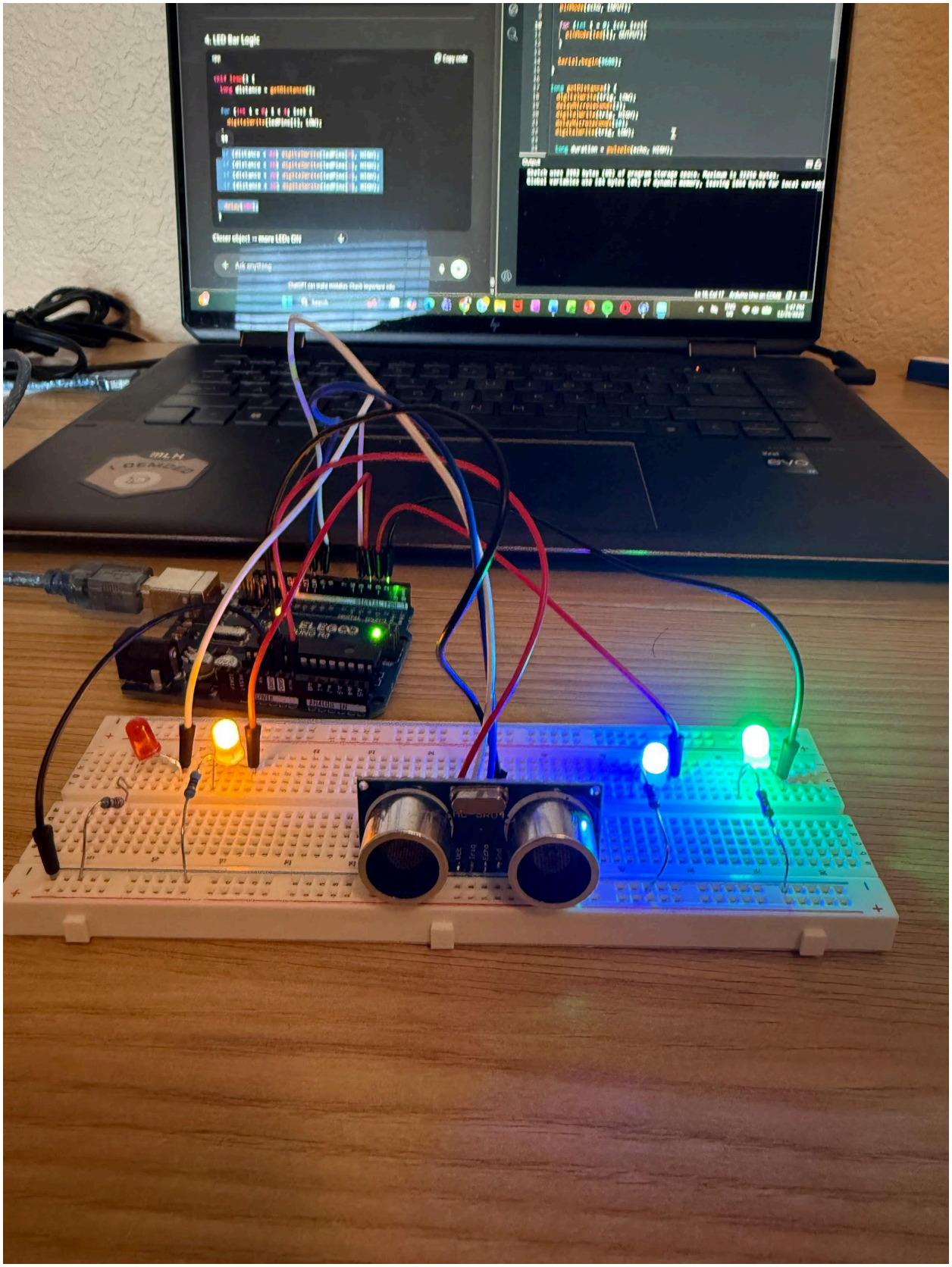


DAY 14 — Object Proximity LED Bar Graph

This project visualized ultrasonic distance measurements using an LED bar graph to represent object proximity. Distance values were quantized into discrete ranges, with each range activating an additional LED as an object moved closer to the sensor. This project demonstrated how sensor data can be transformed into intuitive digital outputs and reinforced concepts of quantization, comparator logic, and real-time visualization used in embedded and hardware systems.

Skills Learned

- Sensor data quantization and thresholding
- Digital visualization using LED bar graphs
- Multi-output GPIO control
- Mapping continuous signals to discrete states
- Real-time embedded system feedback

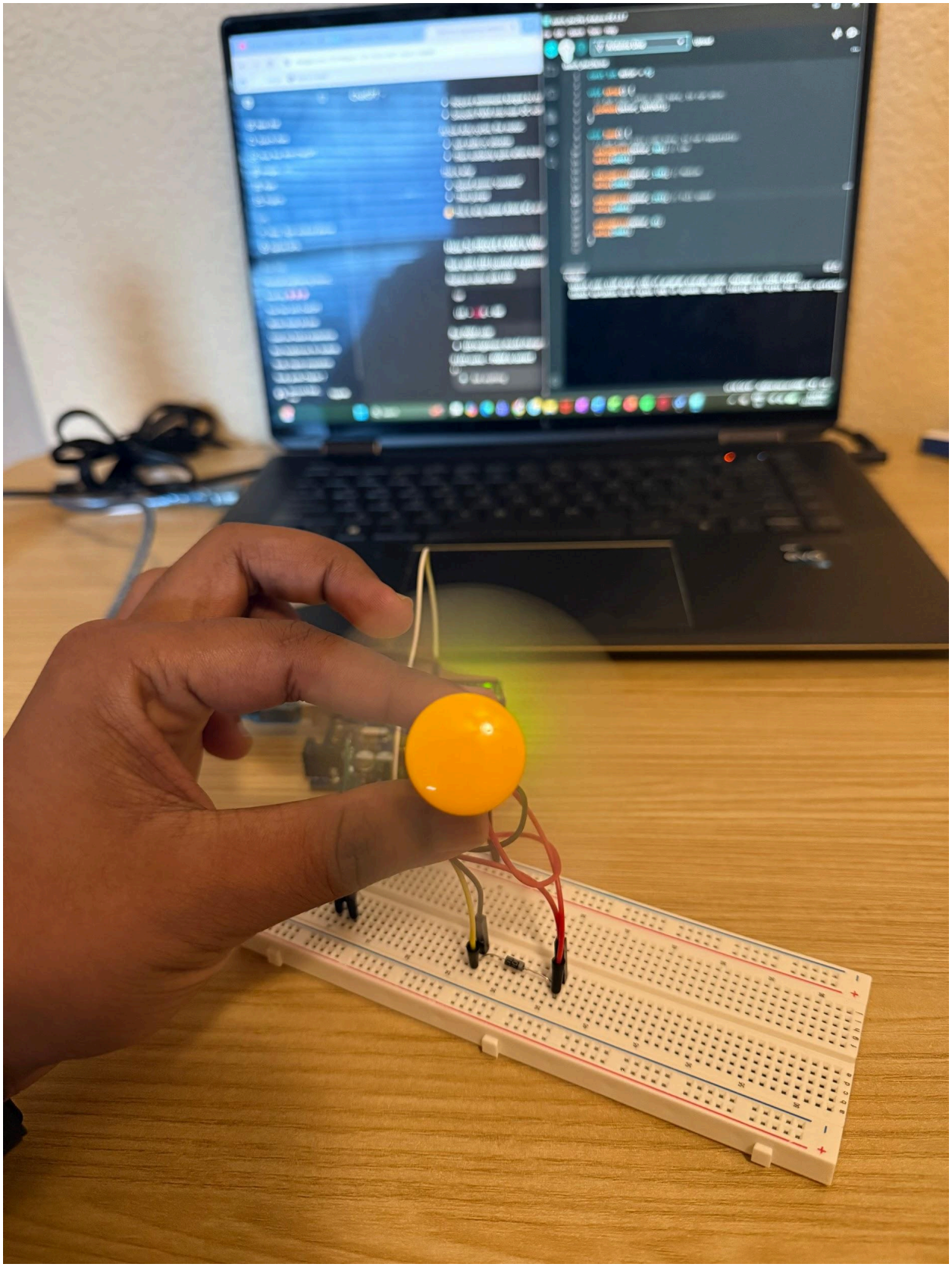

4. LED Bar Logic
Closer object = more LEDs ON
Ask anything
ELEGOO
UNO R3

DAY 15 — Motor Control with Transistor

This project implemented safe DC motor control using a transistor-based driver circuit controlled by PWM signals from the Arduino. By isolating the microcontroller from the motor load and using a flyback diode for protection, the system achieved variable speed control while maintaining electrical safety. This project reinforced principles of power electronics, current amplification, and PWM-based actuation used in embedded and hardware systems.

Skills Learned

- PWM-based motor speed control
- Safe power interfacing using transistors
- Flyback protection for inductive loads
- Hardware safety and current isolation
- Practical power electronics fundamentals



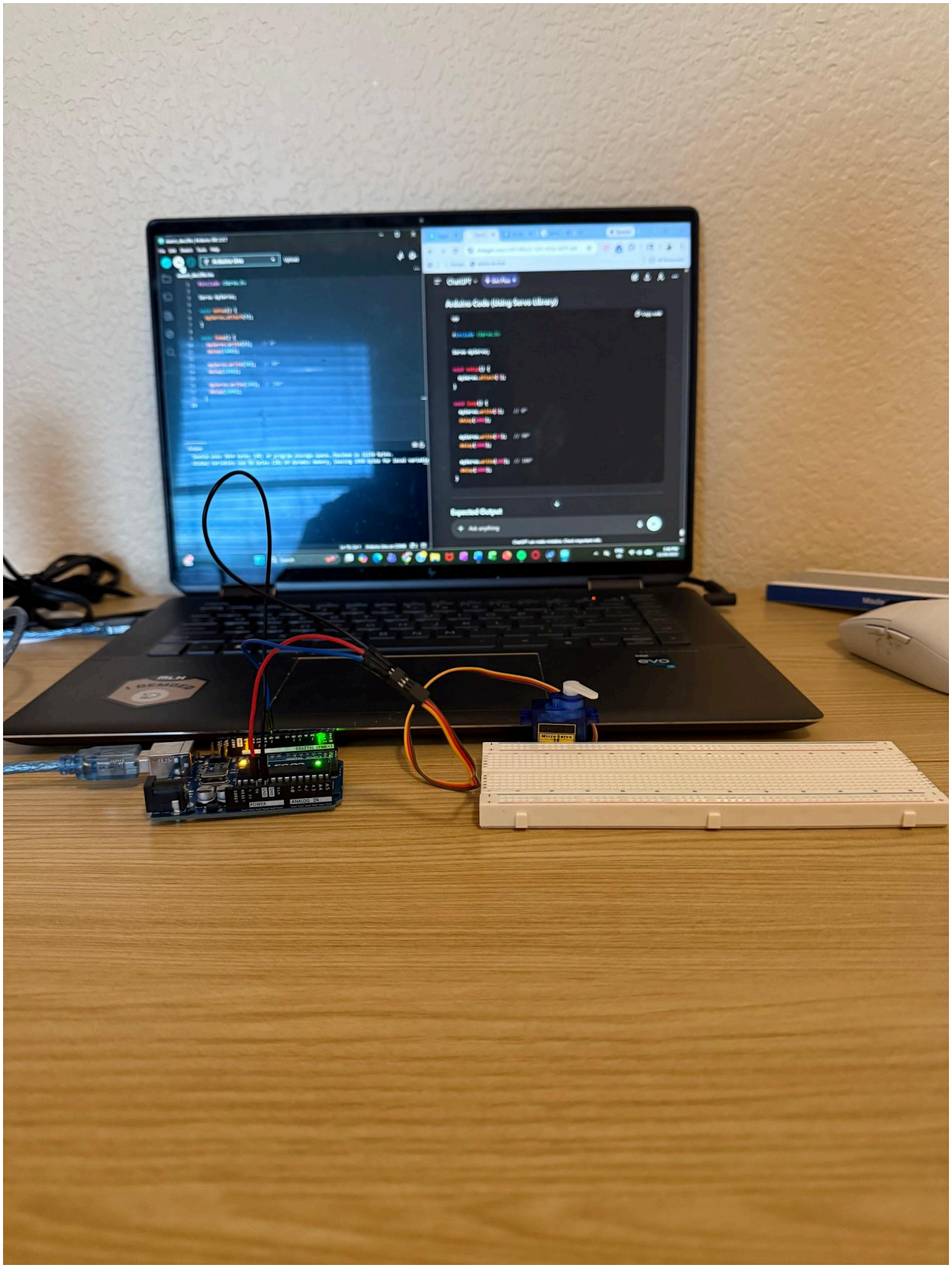
DAY 16 — Servo Motor Precision Control

This project implemented precise angular position control using a servo motor driven by pulse-width modulation signals. Unlike DC motors, the servo interpreted pulse width to determine absolute position, enabling accurate and repeatable movement. The project reinforced closed-loop control concepts and demonstrated how timing precision is used in embedded systems to achieve deterministic mechanical behavior.

Skills Learned

- Precision PWM pulse-width control
- Closed-loop position control concepts
- Servo motor interfacing and power considerations
- Mapping analog inputs to controlled outputs

- Timing-sensitive signal generation

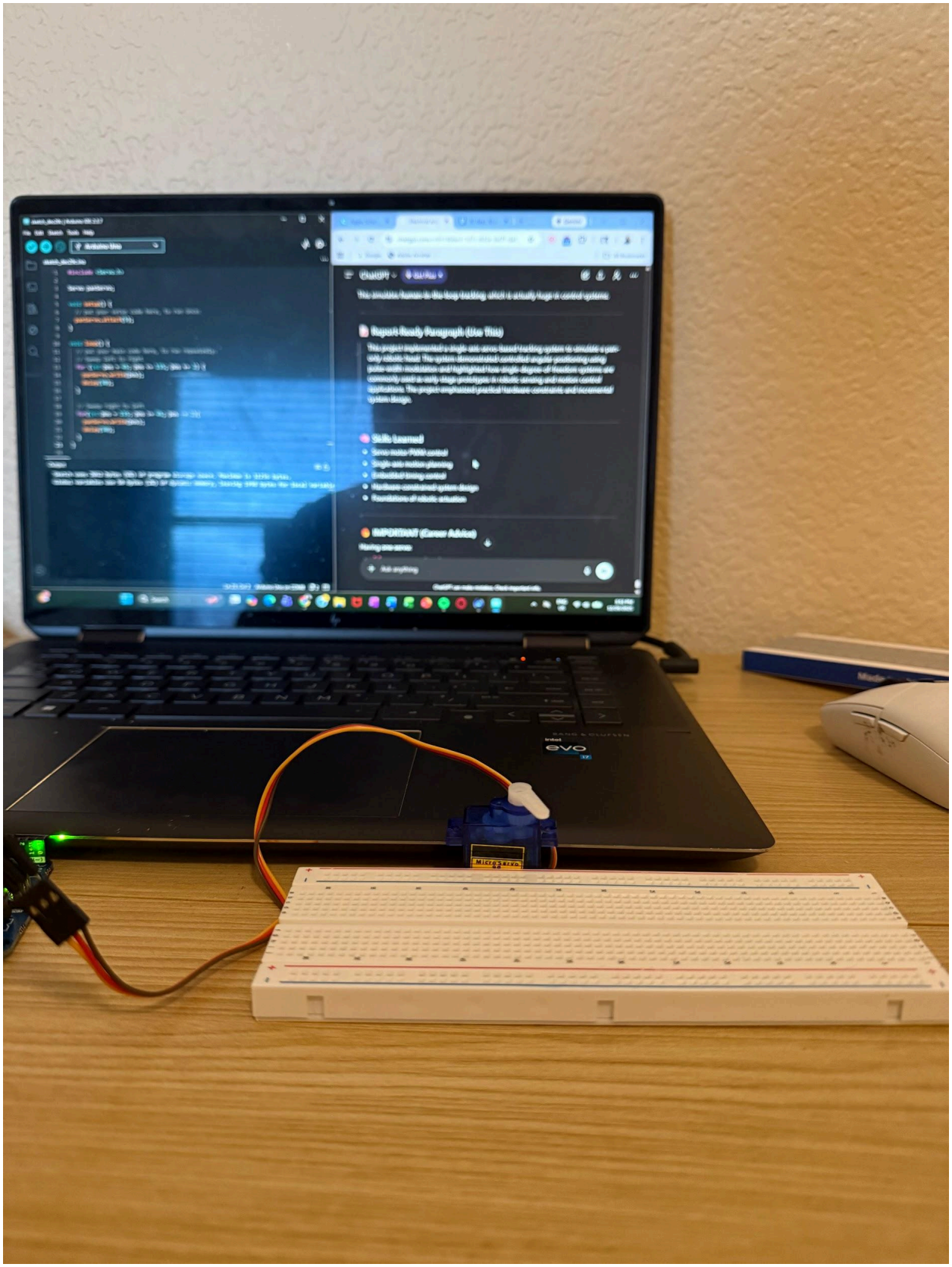


DAY 17 — Mini Axis Pan/Tilt Servo Project

This project implemented a single-axis servo-based tracking system to simulate a pan-only robotic head. The system demonstrated controlled angular positioning using pulse-width modulation and highlighted how single-degree-of-freedom systems are commonly used as early-stage prototypes in robotic sensing and motion control applications. The project emphasized practical hardware constraints and incremental system design.

Skills Learned

- Servo motor PWM control
- Single-axis motion planning
- Embedded timing control
- Hardware-constrained system design
- Foundations of robotic actuation

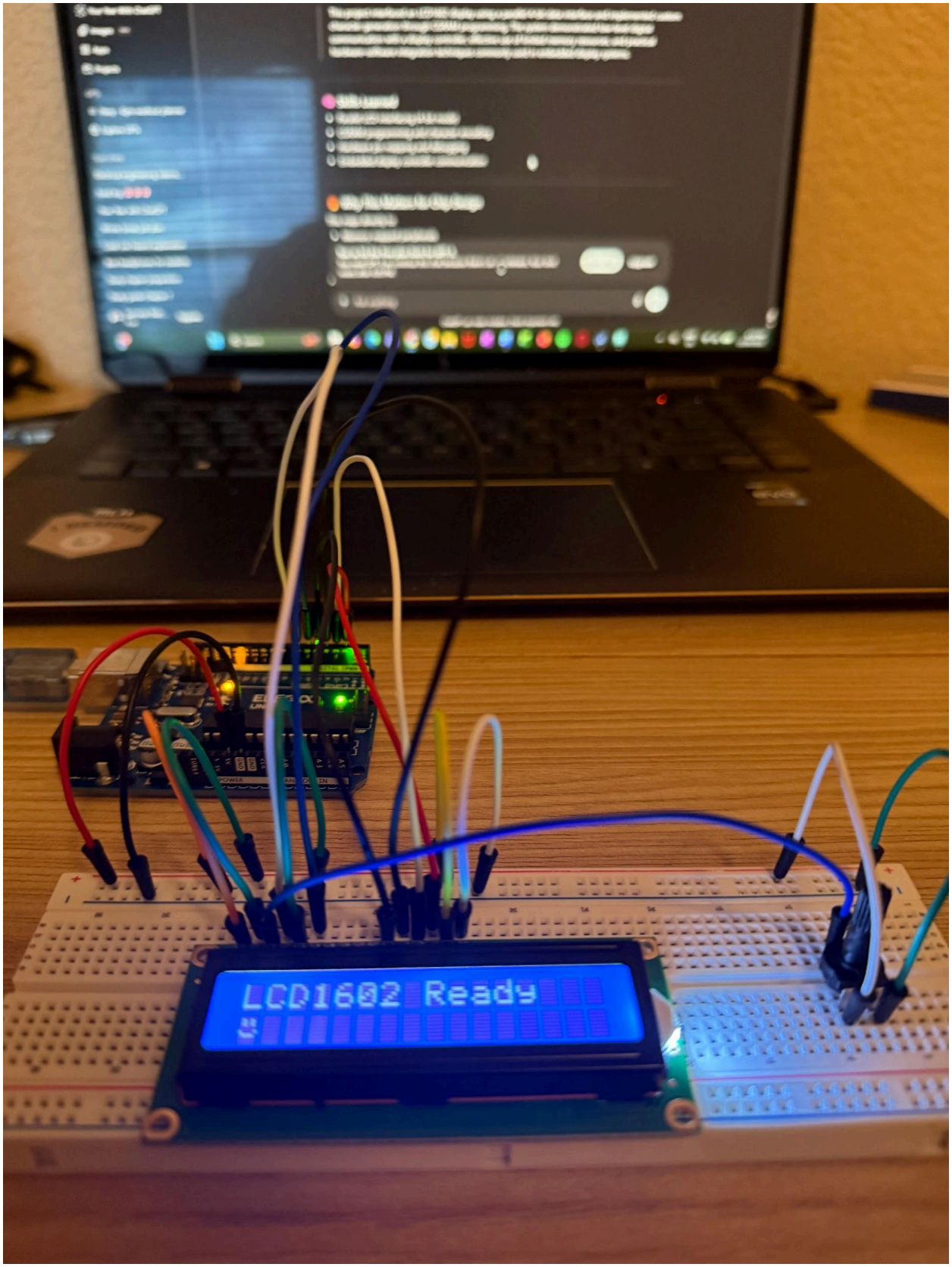

Micro Servo
intel
evo

DAY 18 — LCD Display: Custom Characters

This project interfaced an LCD1602 display using a parallel 4-bit data interface and implemented custom character generation through CGRAM programming. The system demonstrated low-level digital communication with a display controller, effective use of limited memory resources, and practical hardware–software integration techniques commonly used in embedded display systems.

Skills Learned

- Parallel LCD interfacing (4-bit mode)
- CGRAM programming and character encoding
- Hardware pin mapping and debugging
- Embedded display controller communication

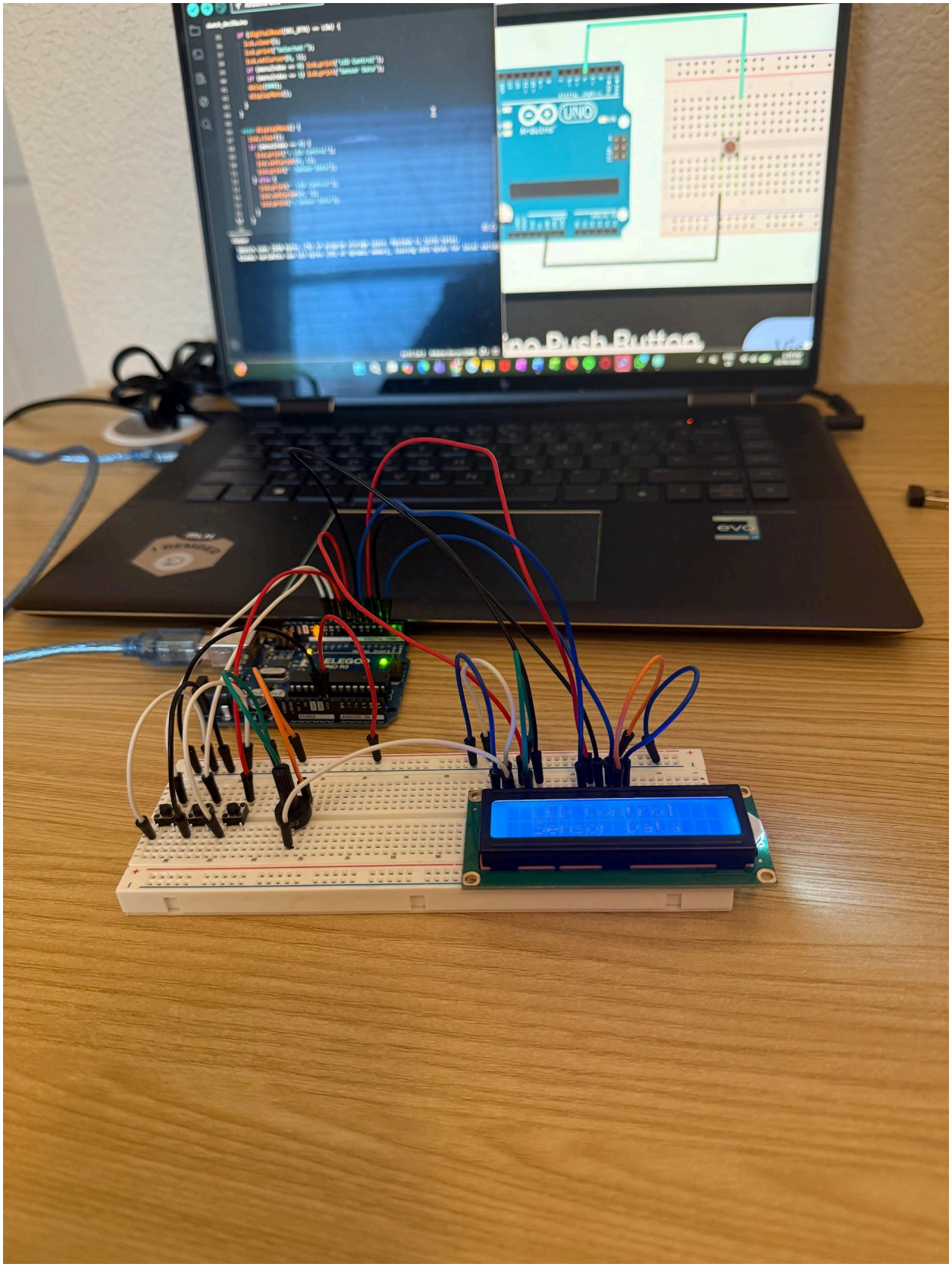

LCD1602 Ready

DAY 19 — LCD Menu Navigation with Buttons

This project implemented a menu-driven user interface on a 16×2 LCD using push-button inputs and a finite state machine architecture. Button inputs were processed using internal pull-up resistors, enabling reliable navigation between menu states. The design demonstrated structured UI control, state transitions, and embedded human–machine interaction.

Skills Learned

- Finite State Machine (FSM) design
- Embedded UI development
- Button input handling with pull-ups
- Menu logic and screen rendering

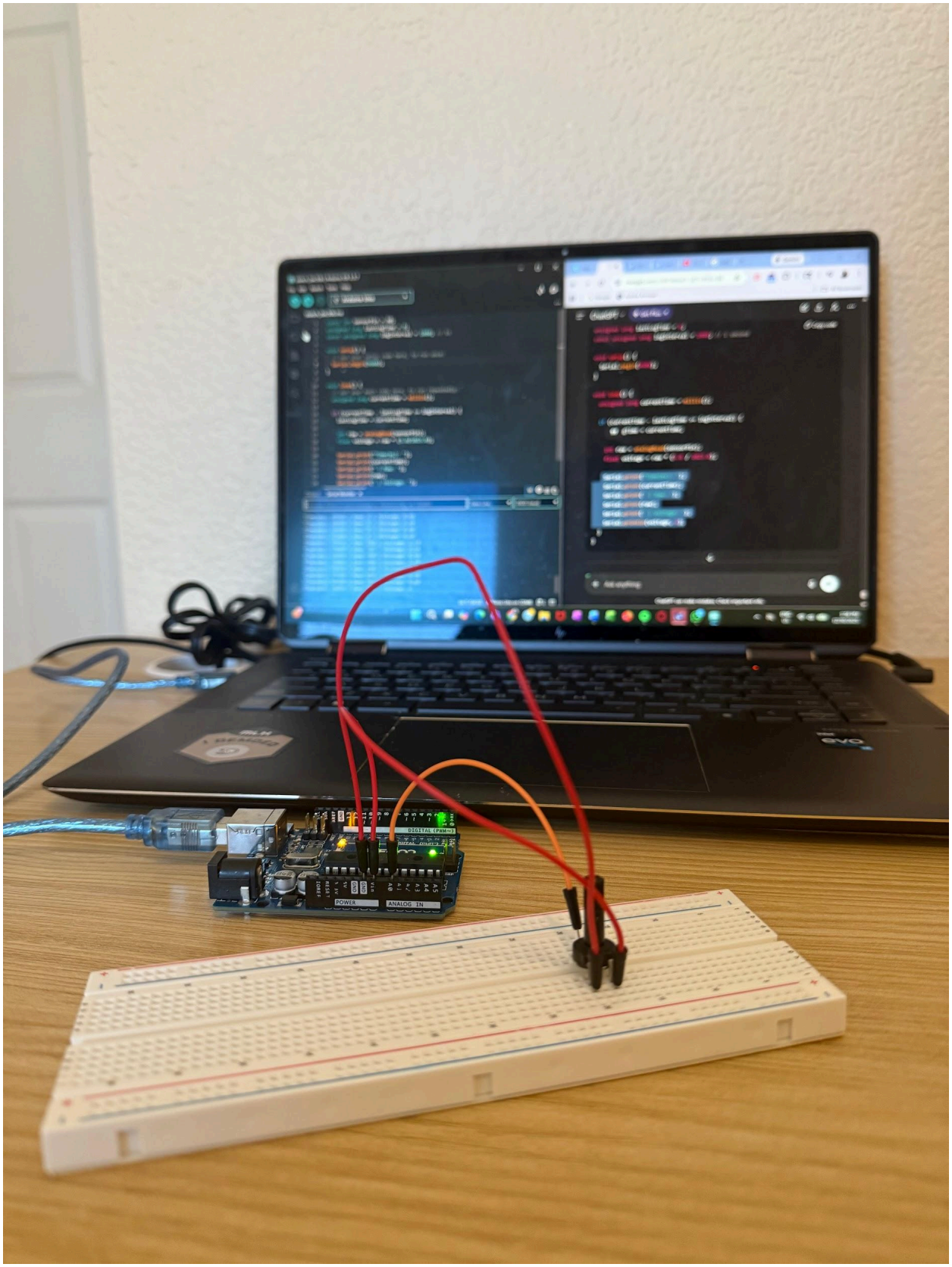

UNO
LED Control
Sensor Data

DAY 20 — Basic Data Logger (Serial Output)

This project focused on implementing a basic data logging system using serial communication. Sensor values were sampled at fixed time intervals and transmitted over the Serial Monitor in a structured format. The project emphasized non-blocking timing using system clocks rather than delays, allowing continuous program execution while logging data reliably. This approach mirrors real-world embedded data acquisition and diagnostics systems, where consistent sampling and traceable output are critical.

Skills Learned

- Periodic data sampling using `millis()`
- Serial communication for real-time logging
- Structured data formatting for analysis
- Embedded systems data acquisition concepts
- Non-blocking program design

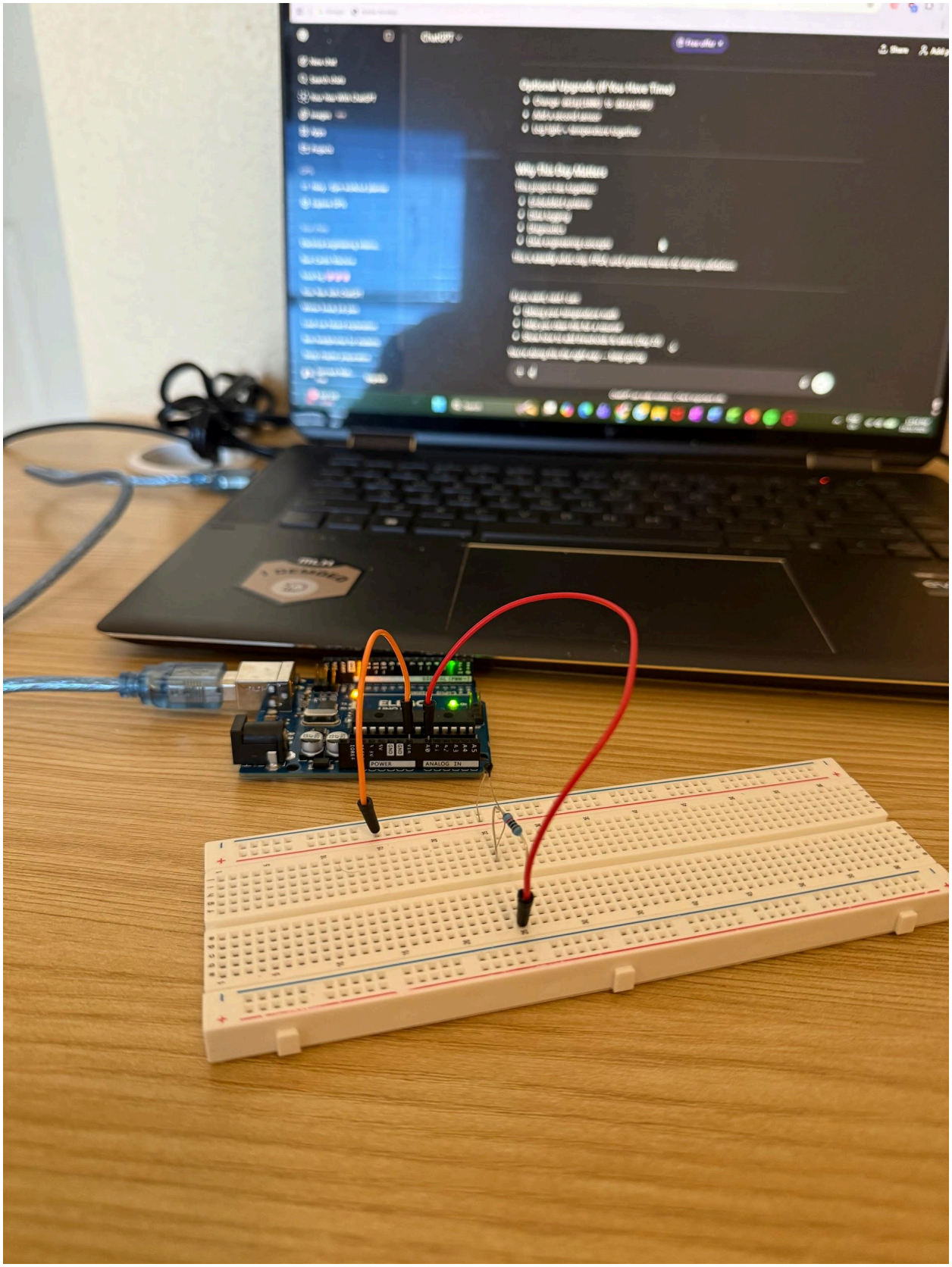

DIGITAL (PWM~)
POWER
ANALOG IN

DAY 21 — Real-Time Temperature Logger to Excel

In this project, I implemented a real-time temperature logging system that outputs sensor data in CSV format via serial communication. Temperature readings were sampled at fixed intervals and formatted to be directly compatible with Excel and other data analysis tools. This enabled real-time plotting and post-processing of embedded sensor data, reinforcing the connection between hardware-level data acquisition and higher-level data analysis workflows commonly used in engineering diagnostics and validation.

Skills Learned

- CSV data formatting for external tools
- Real-time sensor data acquisition
- Serial data logging for analysis and visualization
- Time-based sampling using `millis()`
- Embedded-to-PC data integration

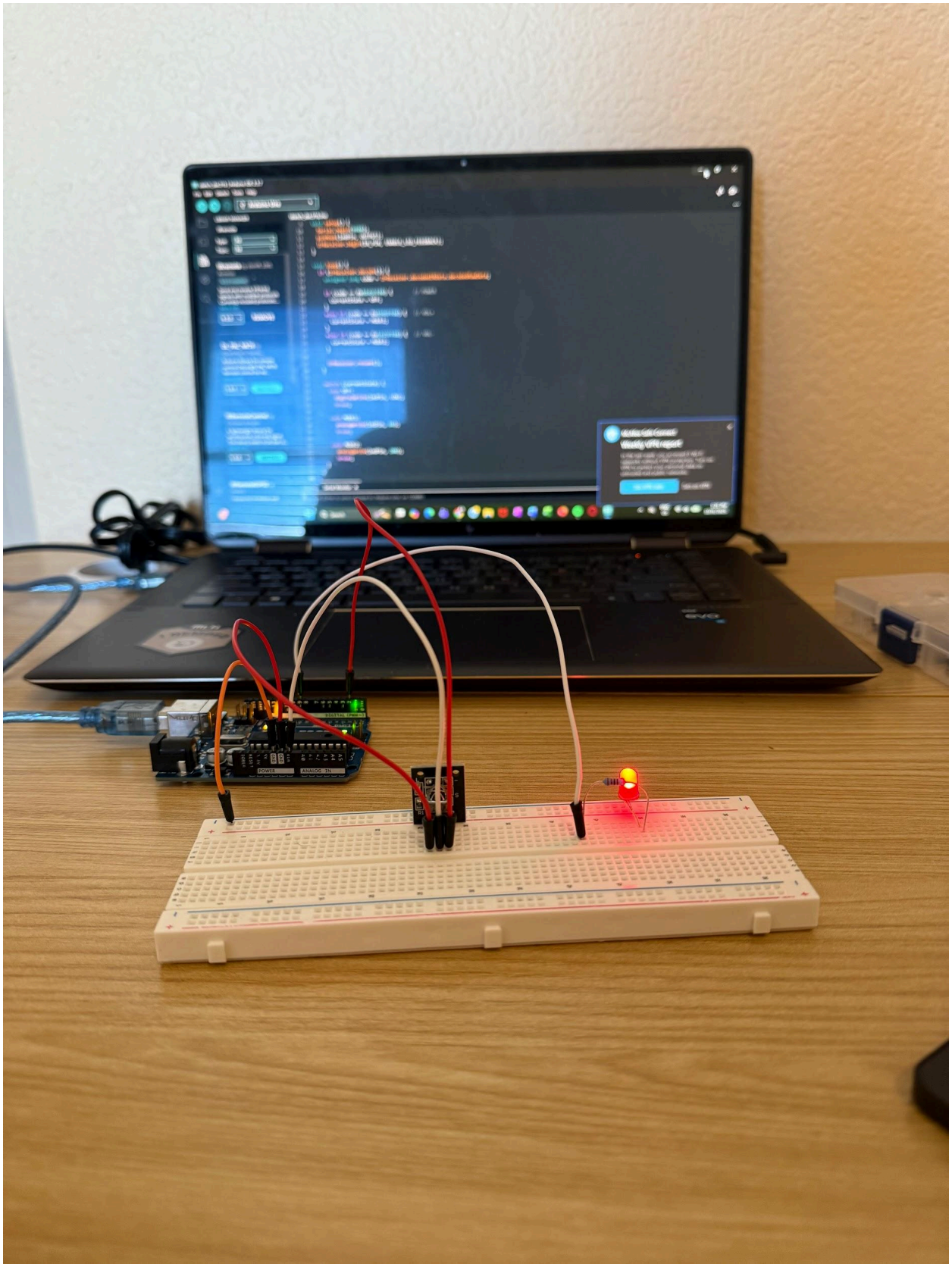

POWER
ANALOG IN

DAY 22 — IR Remote Decoder

This project involved decoding infrared (IR) remote control signals using an IR receiver module and implementing a finite state machine to control system behavior. Distinct IR command codes were mapped to system states, enabling mode-based LED control through PWM. During development, compilation issues related to hexadecimal constant representation were identified and resolved, reinforcing understanding of low-level data encoding and protocol handling in embedded systems. This project demonstrated how external digital inputs can be reliably decoded and translated into deterministic system actions.

Skills Learned

- IR protocol decoding and signal interpretation
- Finite state machine (FSM) implementation in embedded systems
- Use of hexadecimal constants in C/C++
- PWM-based output control
- Debugging compilation and logic errors in Arduino
- Event-driven input handling

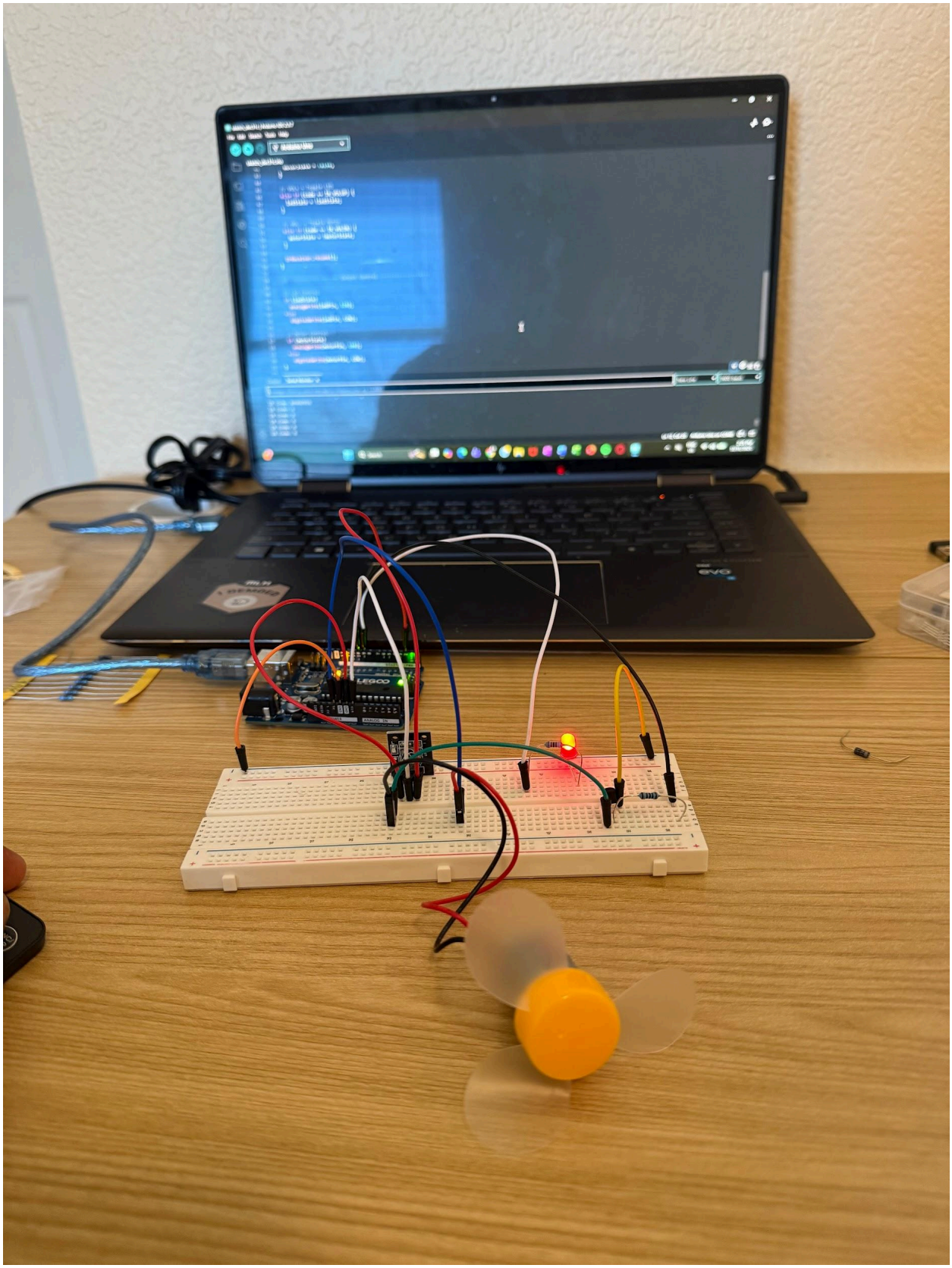


DAY 23 — IR Controlled Motor / LED System

In this project, I designed an IR remote–controlled embedded system that allows a user to control hardware outputs through discrete remote commands. IR signals were decoded and mapped to predefined system states, which in turn controlled an LED and/or motor behavior using digital and PWM outputs. This project emphasized human–machine interaction by translating user inputs into deterministic hardware actions. Building on the previous IR decoding work, this system integrated state-based control logic with real-time actuation, demonstrating how embedded systems interface with users in practical control applications.

Skills Learned

- Human–Machine Interface (HMI) design in embedded systems
- State-based control of motors and LEDs
- Integration of IR input with hardware actuation
- PWM control for variable output behavior
- System-level embedded design and testing
- Debugging real-time input-to-output control logic

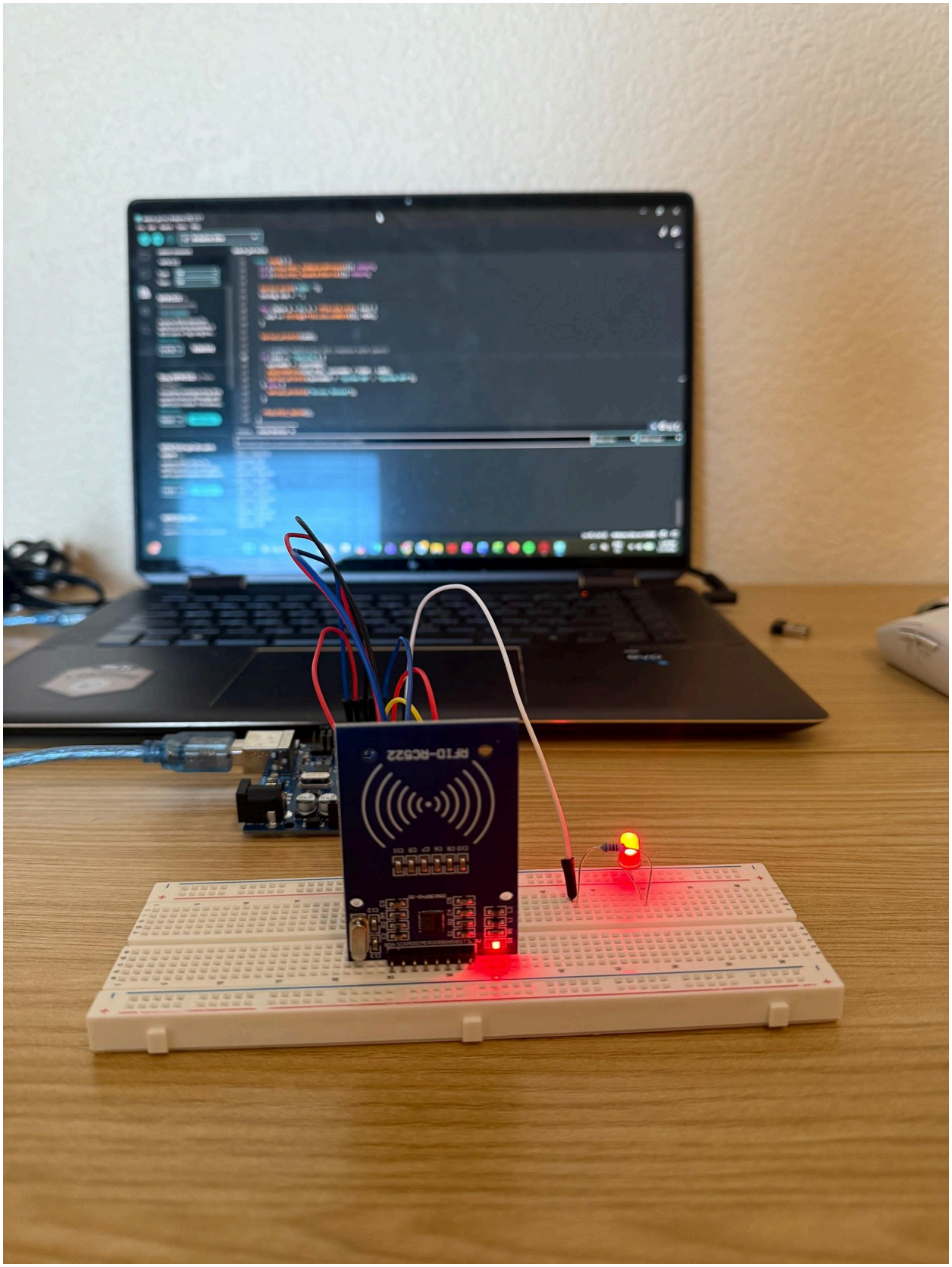


DAY 24 (Adapted) — RFID-Based Access UI (RC522)

In this project, an RC522 RFID module was used to implement a contactless user interface for an embedded system. The RFID reader served as a secure human–machine interaction device, allowing system control through authorized card detection. When a valid RFID tag was scanned, the system toggled its operational state and provided immediate feedback through an LED and serial output. This project demonstrated real-world embedded UI design, SPI communication, and state-based control commonly used in access control and authentication systems.

Skills Learned

- SPI communication protocol
- RFID-based user input systems
- Embedded state-machine control
- Secure access and UID verification
- Hardware interfacing with 3.3V peripherals
- Real-world HMI design concepts

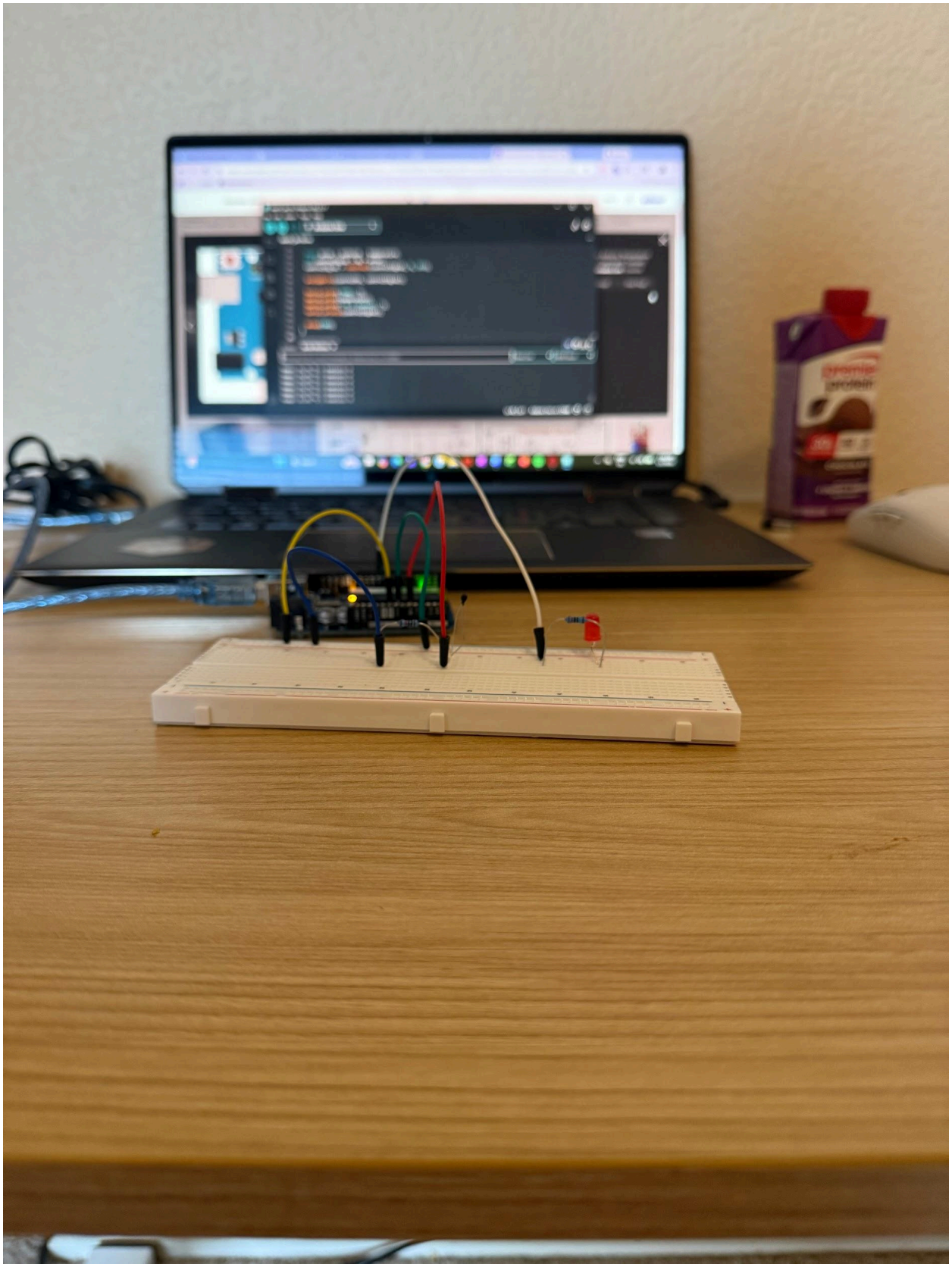

RFID-RC522

DAY 25 — Mini Digital Thermostat

This project focused on designing a mini digital thermostat using a thermistor and PWM-based control. The system continuously monitored temperature and adjusted the output using proportional control logic to maintain a defined setpoint. Rather than simple on-off behavior, the output dynamically responded to temperature error, improving system stability and realism. This project strengthened my understanding of feedback control, embedded system behavior, and real-world thermal regulation concepts.

Skills Learned

- Closed-loop control systems
- Proportional (P) control logic
- Embedded temperature regulation
- PWM-based actuator control
- Sensor-to-control signal mapping
- Debugging feedback instability

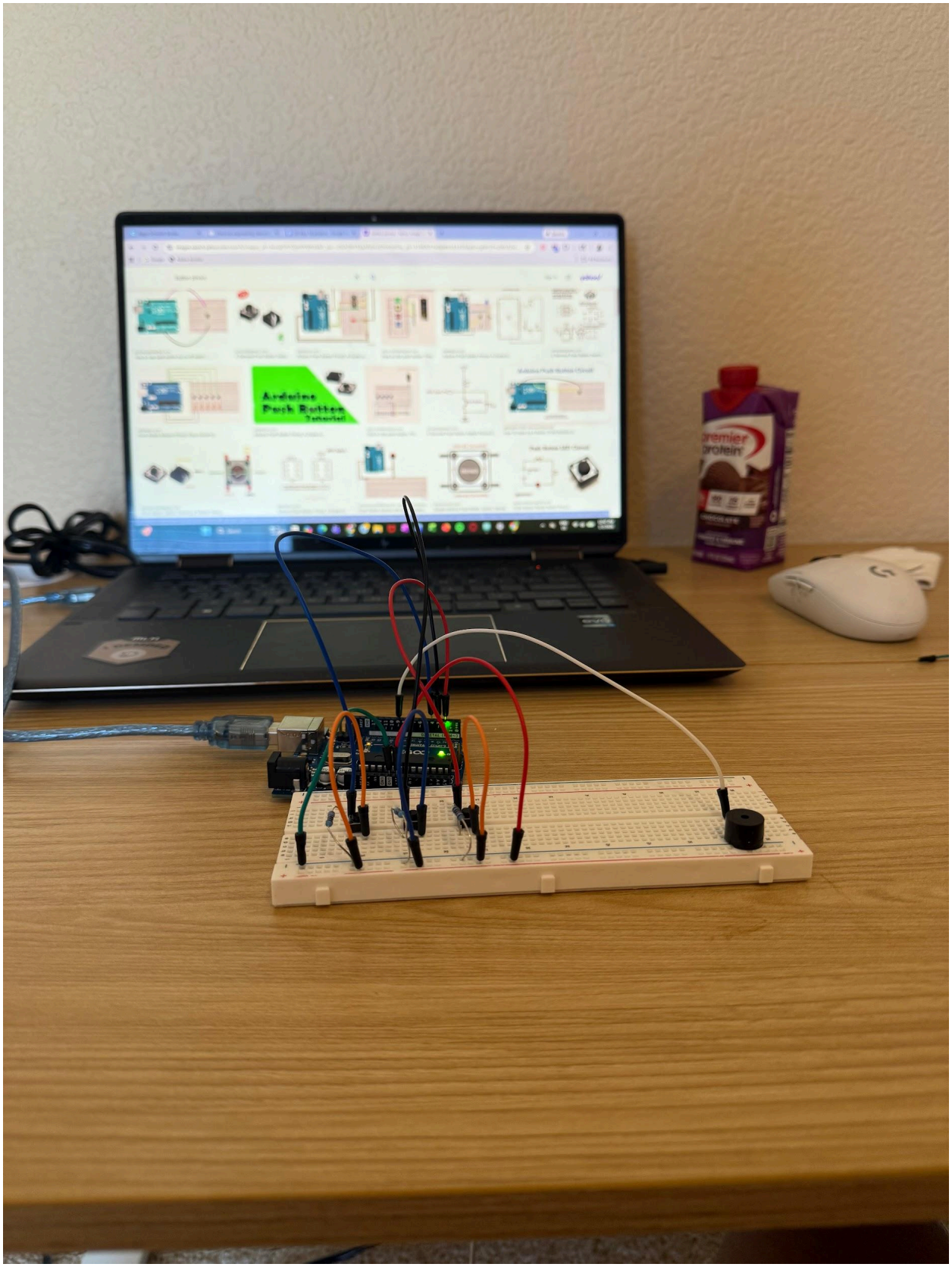


DAY 26 — Audio Tone Synth + Keyboard

This project implemented a simple digital audio synthesizer using push buttons as a keyboard interface. Each button generated a distinct frequency using Arduino’s tone function, demonstrating how digital systems create audio signals through timed square waves. The project reinforced frequency control, user input handling, and signal timing fundamentals.

Skills Learned

- Digital signal generation (frequency control)
- Human–machine interface design
- Timing-based waveform synthesis
- Embedded audio fundamentals
- Button input handling

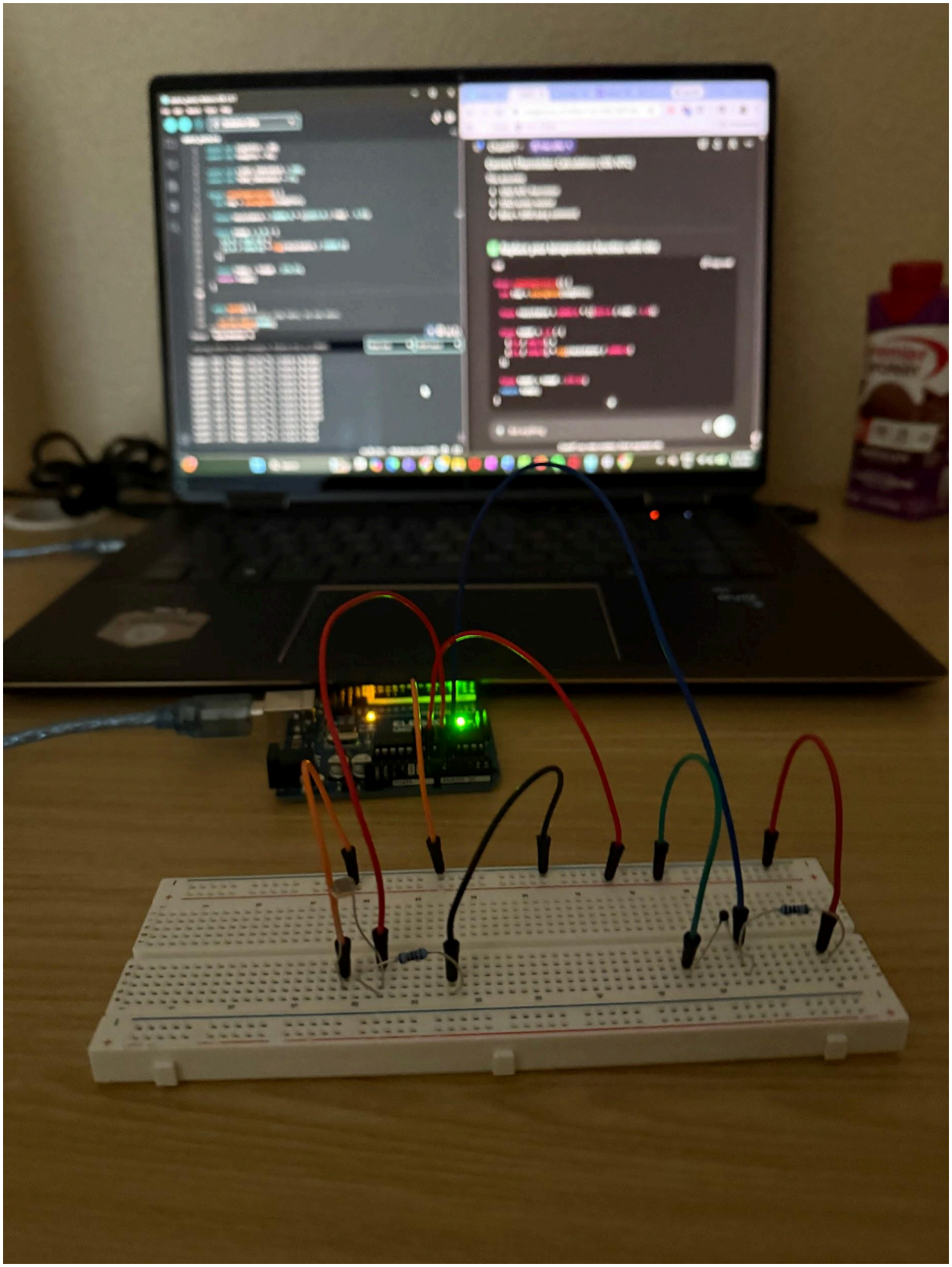

premier
protein

DAY 27 — Tiny ML Intro (No ML Library)

This project implemented a threshold-based classifier using light and temperature sensor data to emulate TinyML inference on an embedded system. By defining decision boundaries and mapping sensor inputs to condition labels, the system demonstrated feature extraction, classification logic, and real-time inference without external ML libraries. The project highlights foundational TinyML concepts executed entirely on-device.

Skills Learned

- TinyML fundamentals (feature-based inference)
- Embedded data classification
- Decision boundary design
- Multi-sensor data fusion
- Real-time inference on microcontrollers

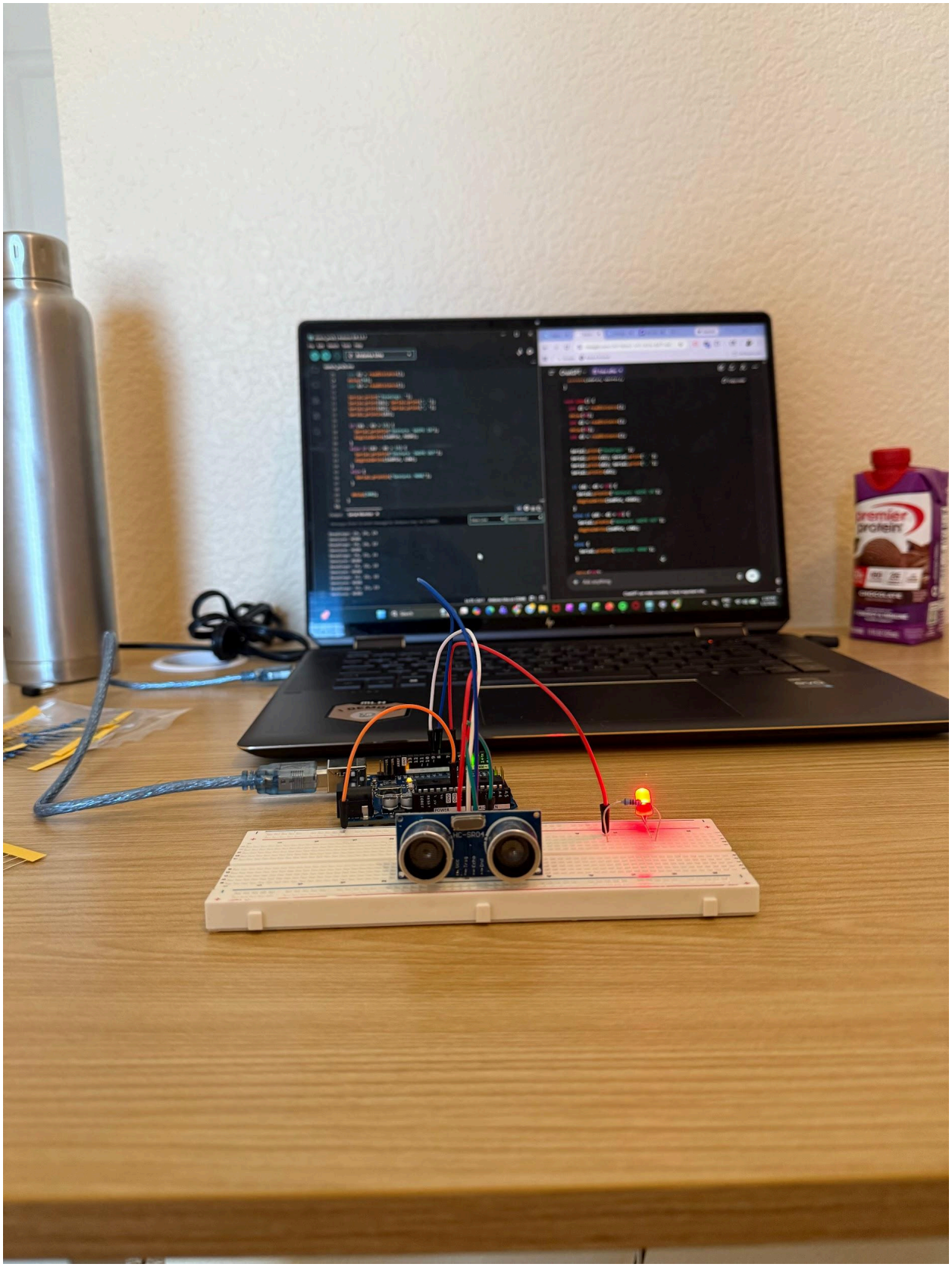


DAY 28 — Embedded Gesture Detection (Acceleration Approx Logic)

This project implemented gesture recognition using ultrasonic time-of-flight measurements. By sampling distance values over short time intervals and analyzing temporal changes, I designed a threshold-based embedded classifier capable of detecting swipe gestures. This approach reflects real-world embedded inference systems where low-power gesture detection is implemented without machine learning libraries.

Skills Learned

- Time-series sensor sampling
- Ultrasonic distance measurement
- Temporal feature extraction
- Threshold-based gesture classification
- Embedded inference logic
- Real-time debugging and tuning


premier protein
HC-SR04

DAY 29 — IoT Simulation

This project simulated an IoT pipeline by transmitting sensor data from an embedded device to a host system over a serial interface. Temperature and light measurements were formatted as structured data and streamed in real time, mimicking cloud ingestion workflows. This exercise emphasized edge-to-cloud data flow, structured telemetry, and system-level thinking.

SKILLS LEARNED

- Embedded telemetry design
- Serial communication protocols
- Sensor data formatting (CSV)
- Edge-to-cloud system modeling
- Real-time data logging
- IoT architecture fundamentals

DAY 30 — Capstone Project: Smart Environment Monitor

This capstone project focused on the design and implementation of a multi-sensor embedded control system that integrates sensing, decision logic, and actuation into a cohesive, real-time platform. The system was architected around a central microcontroller and structured using a state-machine–based control flow, allowing the system to transition predictably between locked, idle, active, and alert states. This approach emphasized robustness, scalability, and clarity in system behavior.

Multiple sensors were interfaced simultaneously, including a thermistor for temperature monitoring, a photoresistor for ambient light sensing, an ultrasonic sensor for proximity detection, and an RFID module for secure access control. Sensor data was continuously sampled and evaluated to drive system decisions. In particular, light intensity was used as the primary control variable, where increasing ambient light

dynamically increased motor speed through PWM control, demonstrating closed-loop feedback and proportional response rather than simple threshold-based switching.

The system incorporated several output devices to visualize and respond to system states. RGB LEDs provided immediate visual feedback for operational modes, a DC motor simulated an active cooling or ventilation response, a servo motor represented mechanical actuation, and a buzzer was used for alert signaling. All outputs were coordinated through centralized decision logic, reinforcing the concept of hardware–software co-design. Additionally, real-time system telemetry was streamed over the serial interface in structured format, enabling offline analysis and forming a foundation for future data-driven or machine-learning applications.

Overall, this project demonstrated end-to-end embedded system design, from sensor interfacing and signal conditioning to control logic, actuation, and diagnostics. The emphasis on modular design, real-time responsiveness, and system integration closely mirrors challenges encountered in modern embedded, chip, and hardware-adjacent engineering roles.

Skills Learned

- Embedded system architecture and modular design
- State machine implementation for reliable control flow
- Multi-sensor interfacing and sensor fusion concepts
- Analog-to-digital conversion and signal interpretation
- PWM-based motor speed control
- Real-time actuator control (DC motors and servos)
- RFID-based hardware access control fundamentals
- Hardware–software co-design and integration
- Structured serial data logging for diagnostics and analysis
- Debugging and validating complex embedded systems