Darius Quick

9/16/25

Module Three Journal

In embedded systems, the way a microcontroller talks to the outside world really comes down to which interface you choose. Three that I’ve worked with or studied are GPIO, PWM, and I2C, and each one has its own purpose.

GPIO is the most basic. A pin can be either on or off, input or output. It’s perfect for simple things like checking if a button is pressed or turning an LED on. The trade-off is that it can’t do much beyond those basic high/low signals.

PWM builds on that by letting you fake analog control using digital pulses. By adjusting how long a signal stays high versus low (the duty cycle), you can dim an LED, control a motor’s speed, or even produce sound. It’s still digital under the hood, but it gives you a lot more flexibility than a plain GPIO.

I2C is different because it’s not just about toggling pins—it’s a communication protocol. With only two wires, you can connect a bunch of devices, like sensors or displays, each with its own address. That saves pins and wiring complexity. It’s not as fast as something like SPI, and it takes more effort to set up, but for systems with multiple peripherals, it’s a huge advantage.

When I think about when to use each, it’s really about the job at hand. GPIO is fine when you just need a simple signal. PWM makes sense when you want variable control. And I2C shines when you must talk to several devices without eating up all your microcontroller’s pins. Each one has its strengths, and knowing which to use is part of a good, embedded design.