Jordan Mitchell

1/22/2025

Professor Moscon

Journal: Peripheral Interfaces in Embedded Systems

*Embedded systems communicate with the outside world via their peripherals. In this journal assignment, you will compare and contrast various interfaces and explain why you would use one over another. Research various interfaces to learn about the differences between communications protocols in embedded technologies, their uses, strengths and weaknesses. Describe the differences between any three: GPIO, PWM, A2D, SPI, I2C, and UART. Why would you use one interface over another? Explain the differences between the three interfaces chosen. Explain why you would use one interface over another.*

Embedded systems rely on various interfaces to connect with the outside world. Whether it’s turning on the power, controlling a CPU fan, or communicating with another device, these interfaces play a major role in making everything run smoothly. Among the many options, GPIO, PWM, and UART stand out as some of the most commonly used. Each serves their own unique purpose with strengths and limitations.  
  
**General Purpose Input/Output (GPIO)**

GPIO is one of the simplest and most straightforward interfaces. It's a versatile digital interface that's used to control or monitor external devices. It can be configured as an input to read signals, like those from a button, or an output to send signals, like turning a light on or off. GPIO is simple to use and very flexible, making it an ideal choice for basic on/off controls in embedded systems. The downside is, its functionality is limited to digital signals, lacking support for complex or analog signals.

**Pulse Width Modulation (PWM)**

PWM, on the other hand, is designed for more precise control. It’s used to control things like the brightness of LEDs or the speed of motors by rapidly switching between high and low states. The amount of time the signal stays “on” versus “off”, known as the duty cycle, determines the device's output. For example, a higher duty cycle means a brighter LED. PWM is more sophisticated than GPIO and great for situations where you need fine adjustments, but it does require precise timing and hardware support.

**Universal Asynchronous Receiver Transmitter (UART)**

UART is a bit different. It’s a serial communication protocol meant for sending data between devices. This could mean connecting your embedded system to a GPS, a Bluetooth device, or even another microcontroller. While UART isn’t the fastest communication protocol, it’s reliable and easy to set up, making it a popular option for applications where high speed data transfer isn’t a priority.

**Uses and Comparison**

* Communication: GPIO is suitable for binary on/off signals, while UART handles serial data communication. PWM is best for controlling devices centered around analog outputs.
* Ease of Use: GPIO is the easiest to use whereas PWM requires a bit more configuration. UART is simple but requires understanding of serial communication basics.
* Applications: GPIO is ideal for simple on/off tasks, like toggling LEDs or switches. PWM shines when you need precise control, like adjusting the brightness of an LED or the speed of a fan motor. UART is the best choice when you need to send data to external modules or sensors.
* Complexity: GPIO is the most basic, being the easiest to implement. This is followed by PWM, which needs precise timing. UART requires data synchronization. The latter two are more advanced and designed for specific tasks.

**Final Decision**

The decision to use GPIO, PWM, or UART ultimately comes down to your project's requirements. If you’re working on a simple task, like turning on lights, GPIO is all you need. If the project requires fine tuning or variable levels of control, PWM is the way to go. When it’s time to connect your system to another device, UART is perfect for getting the job done. Each interface has its place, and understanding their strengths makes it easier to pick the right one for the job.