**Interfaces Abound!**

**CS-350**

Damean Murphy-Short

Southern New Hampshire University

January 20, 2025

This paper aims to discuss a number of interfaces through which digital communication can take place. Among the options presented to us for discussion are: GPIO, PWM, A2D, SPI, I2C, and UART.  
 Note from the author: I find it necessary to discuss, before moving any further, that PWM and A2D are not interfaces. A2D, as far as I can tell, unless I hold some misconception, stands for Analog to Digital, which does not have a specific interface or standard which we can compare to the other interfaces listed. Whereas the other options transmit and receive data (primarily serially), analog-to-digital-conversion samples and converts a signal, rather than perform any messaging. Pulse-width modulation is also not an interface, but rather a type of signal that may be transmitted on an interface, if I understand the term correctly. Briefly, pulse-width modulation is the controlled repeated application of voltage over a fixed period. With that out of the way, we can more readily discuss the other interfaces.

The above interface standards can largely be separated into the categories of synchronous serial communication or asychronous serial communication, with the exclusion of GPIO, which can be used as serial, parallel, synchronous, or asynchronous, depending on the application chosen. Because GPIO is a flexible assortment of pins (general purpose input and output), it can communicate with any of the other listed interfaces when programmed properly, or used as a single-purpose interface for whichever sensors or devices one wishes to connect. GPIO could also be used to simply provide or detect voltage when desired (represented as a single bit), without reading any bytes of data.

The synchronous interfaces listed are so-named because they synchronize messaging using a clock signal. I2C and SPI each dedicate a single wire as a serial clock. The asynchronous interface, UART, requires the receiver and transmitter to have a shared configuration of baud rate, stop bit size, data bit size, and whether or not a parity bit is included in each frame. In this way, data can be sent serially with a fixed frame-size prepended and appended by a start and stop bit, respectively.

Physically, the interfaces are all fairly similar, but differ based on their specific needs. I2C specifies 2 wires to make a connection and UART requires at least 2 wires (I do find any specification that limits the interface to 2 wires), but I2C only uses one wire for transmission and reception of data (the other is used as a clock, as mentioned above), whereas UART uses a single wire for transmission and another for receiving data. SPI, however, uses four wires: a clock, a ‘chip select’ wire, a transmission (called MOSI), and a receiving wire (called MISO). The chip select wire raises another point about the physical connections that separate the I2C and SPI interfaces from UART: I2C and SPI allow for multiple controller and multiple target/subsidiary devices on a bus. UART transmission does not specify an intended recipient for transmitted data, and so any device connected to the bus with proper configuration will read the data.

The three interfaces all share a common goal in serial communication, but their differences lend them strengths in different applications. For instance, if multiple devices are indeed going to be communicating over the same bus, I2C or SPI would be desireable choices over UART, which does not support device addressing natively (theoretically, this could be done in software). Because UART sends data in simple frames and does not use additional wires to send commands or synchronize to other devices connected to the interface, it is more easily emulated in software than I2C or SPI might be. On the other hand, I2C uses acknowledgement signals to ensure each byte is received from the transmitter, and so might be a better choice when it is vital that each byte is received in order and acknowledged (again, this could be possibly managed in software with the other interfaces). Overall, each interface has its place, depending on the task required of the device being developed.