William Geiger

CS-350

1/28/2024

Module 3

Journal

Within this journal I will be comparing three different embedded system interfaces. I have decided to research GPIO, I2C, and SPI. UART (Universal Asynchronous Receiver Transmitter) is one of the most used serial protocols. It uses a single data line for transmitting and one for receiving data. With UART most often 8-bit data is transferred by 1 start bit (low level), 8 data bits and 1 stop bit (high level). The low-level start bit and high-level stop bit means there is always a high to low transition to begin communication. Microcontrollers which are set to communicate via UART must agree on the bit-rate because they only have the star bits falling edge to synchronize. This makes UART not reliable for long distance communication. The timing dependency is a big drawback for UART.

SPI (Serial Peripheral Interface) is another simple serial protocol. It uses a master to send a clock signal and on each clock pulse it shifts one bit out to the servant and returns one bit back to the master. SPI uses the signal names SCK for clock, MOSI for master out servant in, and MISO for master in servant out. By using SS (servant select) signals the master can control more than one servant on the bus.

I2C (Inter-Integrated Circuit) is also a synchronous protocol. While the previous two examples simply shifted bits in and out, I2C is a bit more intelligent. I2C uses 2 wires, one for the clock and one for the data. This means that a master and servant send data over the same wire, controlled by the master which creates the clock signal. I2C doesn’t use separate servant selects to select a particular device but has addressing features. The first bit sent by the master has a 7-bit address and a read/write bit, which indicate whether the next bytes will come from the master or the servant. After each byte the receiver must send a “0” to acknowledge the reception of the byte. This lets the master choose whether to write or receive bytes as needed.

Each of these interfaces are useful in their own ways. If I was making a simple device that only needed to receive or send a single signal at a time I would probably choose a UART interface. This is a simple interface but for something like a doorbell, you don’t really need anything very complex. If I was making a device that needed to send and receive signals to and from multiple areas I would probably use an I2C interface. This would make more sense for something like a proximity sensor where multiple small sensors are waiting to signal a master controller which will then accomplish some action when it receives a signal.