# 5. An SPI Device Driver for LED Strip and Accurate Delays in Linux

**key words**: SPI device driver, synchronous / asynchronous transfer, SPI-MOSI, hrtimer, rdtsc, ndelay / mdelay, bit banging.

**Approach 1: SPI-based approach:**

- Implement a SPI device driver which can accept pixel information from users and light up 16 RGB LEDs of a LED ring integrated with WS2812 drivers.
- The LEDs of the ring are connected serially and use a 1-wire communication protocol to receive pixel data for full color display (8 bits for each R, G, and B color).
- The data transmission speed is around 800Kbps.
- The ring should be registered as a SPI device of Galileo Gen 2 board by the probe function of the WS2812 driver once a match of device and driver is found.
- The following device file operations:

  *open:* to open a device (the device is "/dev/*WS2812*").

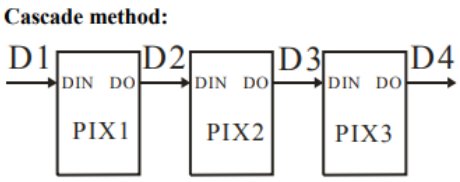
  *write:* The write call sends *n* pixel data to light up the *n* LEDs of the ring, where *1≤n≤16*. Each pixel data consists of 3 bytes for green, red, and blue colors. The call initiates an asynchronous SPI transfer that leads to a proper 1-wire data transmission on SPI-MOSI. As a consequence, the WS2812 can receive the display data and drive the LED display accurately.

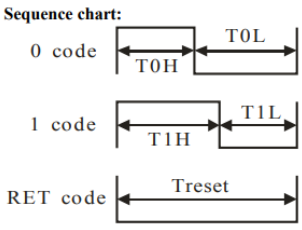
  *ioctl:* to include one command "RESET" to reset the SPI to a suitable operation mode. Note that the SPI mode must be defined properly to make correct 1-wire data transmission. The IO pin multiplexing should also be reset to enable the connection from SPI_MOSI to Data_in of the ring.

  release: to close the descriptor of an opened device file.

**Approach 2: bit banging**

- Using software and delay loops to control a gpio output signal to generate any waveform.
- For instance, to generate a pulse of 5µsec, a program can write 1 to a gpio pin, wait for 5µsec, and finally write 0 to the gpio pin.
- This approach is highly dependent of the accuracy of delay functions and any possible preemptions.
- Do experiments to verify that the timing requirement of WS2812 can / cannot be met by any of the two approaches: ndelay() and hrtimer.




**Data transfer time**( TH+TL=1.25µs±600ns)

| | | | |
|---|---|---|---|
| T0H | 0 code ,high voltage time | 0.35us | ±150ns |
| T1H | 1 code ,high voltage time | 0.7us | ±150ns |
| T0L | 0 code , low voltage time | 0.8us | ±150ns |
| T1L | 1 code ,low voltage time | 0.6us | ±150ns |
| RES | low voltage time | Above 50µs | |