ASSIGNMENT-3 PART-2 BIT BANGING REPORT

TEAM-4

Aditya:1210903678, Dheemanth:1213381271

Following are the three approached we tried for bit-banging to lit up LEDs on WS2812:

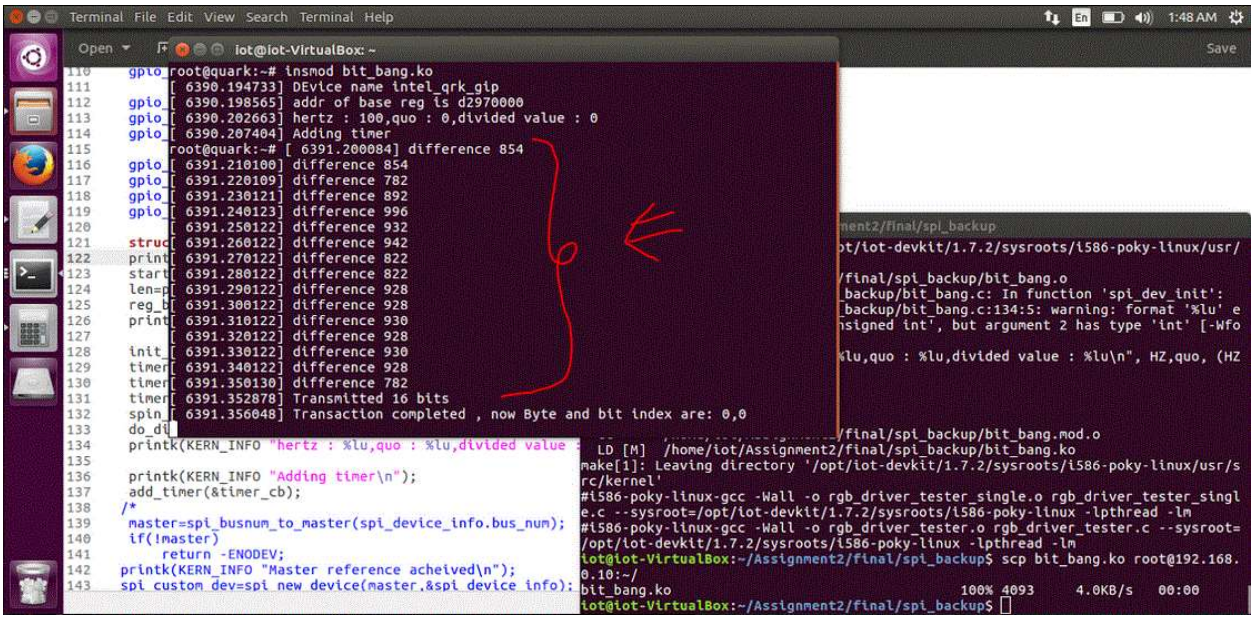
1.) **Setting up gpio pins using "gpio_set_value()" function from <linux/gpio.h>:**
This approach didn't work because gpio_set_value was taking a long time (as compared 0.35microseconds which is the minimum time needed for encoding 0 bit for WS2812). Therefore, instead of setting up gpio values using gpio_set_value function we tried writing memory mapped I/O registers directly which leads to our next two similar kind of approaches.

2.) **High-Resolution timer:**
In this approach we are directly writing the memory mapped IO registers for gpio12 pin:

```
if (global_counter % 2) {
    iowrite32(read_value & (~BIT(4)), reg_base);
} else {
    iowrite32(read_value | (BIT(4)), reg_base);
}
spin_unlock(&lock);
```

Using hrtimer, we are keeping our gpio high/low for the time mentioned in the specification for encoding 0 and 1 bit. We tried setting all the LEDs to common color(red) in circular manner but due to some more delay than expected in using the hrtimer function calls we are not able to set the color properly.

Then we tried taking tsc timestamps before and after a single iowrite32 to corresponding register for gpio12 and we found the difference to be very large than required. Following is the screenshot for this activity's output:



Above screenshot shows that we are getting average tsc difference of around 880, which shows that single iowrite32 is taking is around 880*2.5 = 2200ns (400Mhz CPU speed) which is way more that than we require but we can't rely on this information because computing tsc itself might take a lot of time.

Hence we started bit-banging to gpio12 using hrtimer able to glow LEDs but white in color instead of red(expected) in circular manner, by sending data for a single pixel to all 16 pixels.

3.) Using ndelay():

This approach is basically similar to hrtimer approach in which we are keeping the gpio pin to high and low after setting and unsetting by writing to it's corresponding memory mapped register, but this are the delay between setting and setting is given using ndelay() function.

```
    /*sets the bit at 4th position which represents data
@ gpio 12 to high or low depending if global counter is
even or odd*/
    if (global_counter % 2) {
        iowrite32(read_value & (~BIT(4)), reg_base);
    } else {
        iowrite32(read_value | (BIT(4)), reg_base);
    }
    spin_unlock(&lock);
    /*Sets the delay such that data is latched for a specific
amount of time to glow the led strip as per its data bit encoding standard*/
    ndelay(var_data[global_counter++]);
}
```

Above screenshot shows the use of ndelay() function. We have created a separate kernel module with ndelay implementation. Steps for execution for this module is already mentioned in the readme file for part2.

After executing this module, we are able to glow all the LEDs in circular manner but the problem remains the same that we are not able to control the color of our LEDs. Every pixel is glowing to bright white light only.

Note: Only difference that we observed between hrtimer and ndelay() approach is that sometimes even after banging bits for all 16 pixels we are able to glow only 14 pixels in case of hrtimer, but all 16 using ndelay() always.