WS2812b LED controller with 16x2 LCD display

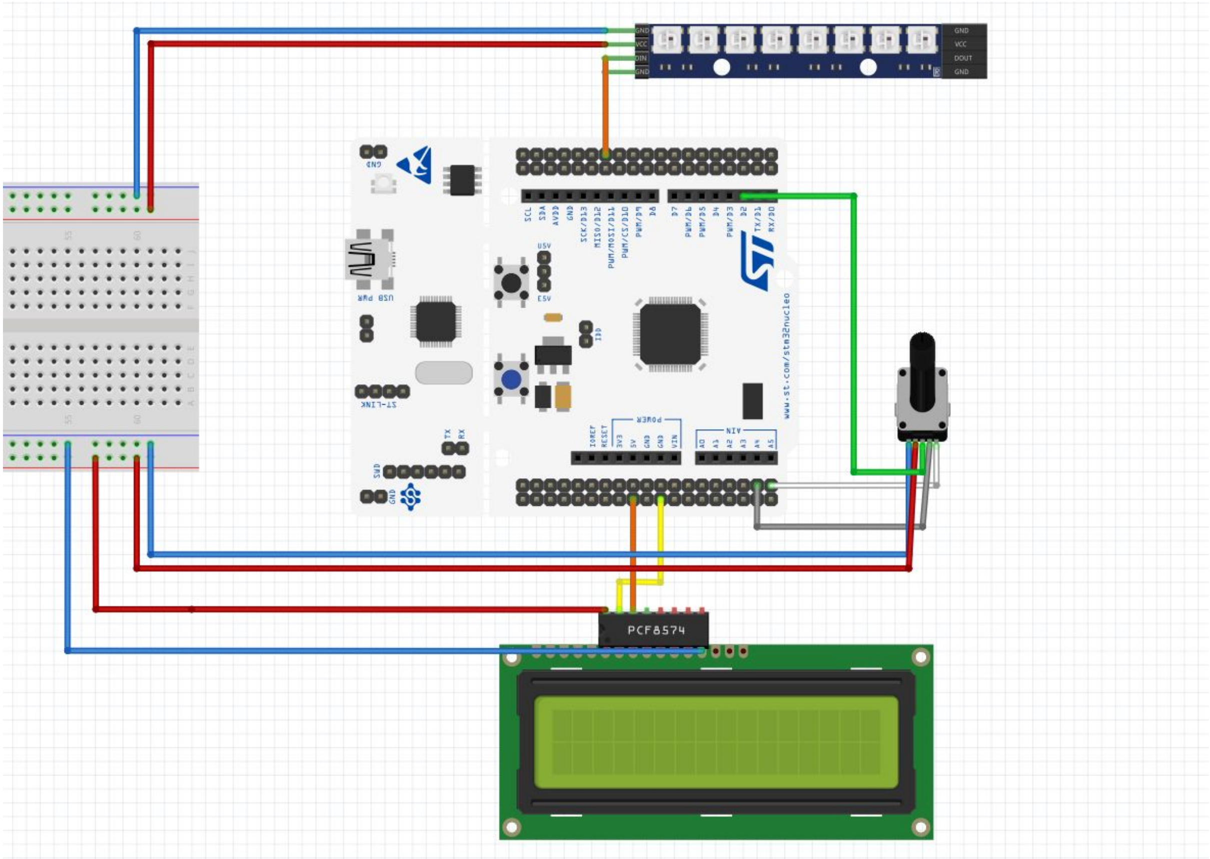
Previously submitted specification

- The main goal of the project is to create a controller of led strip by encoder. Additional is a LCD display which can show a small menu.
- reading values from encoder, create menu with LCD display, control color of LED strip.
- development board Nucleo f303, encoder, 16x2 LCD display.

Hardware description

Project include:

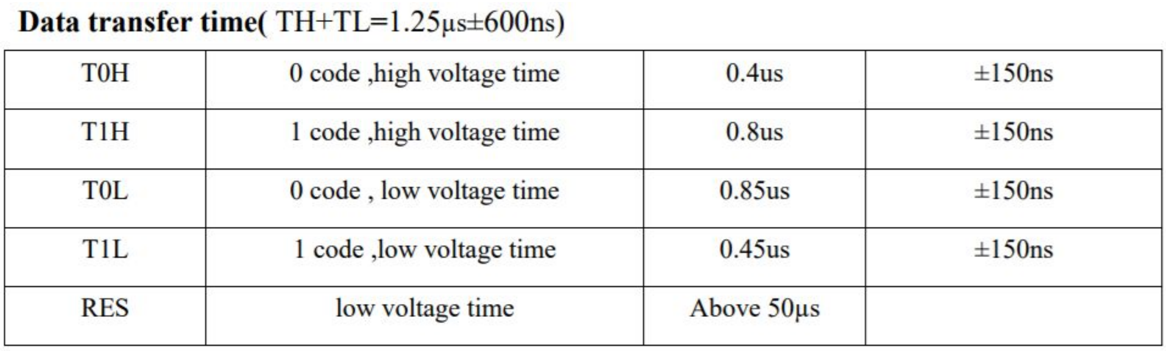
- Nucleo F303RE
- WS2812b led strip
- rotary encoder with button
- 16x2 LCD display with i2c converter PCF8574AT



Software description

WS2812

At first I checked a ws2812b led datasheet to realise how to configure a clock of SPI transmission.



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

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.8us	±150ns
T0L	0 code , low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.45us	±150ns
RES	low voltage time	Above 50μs	

Byte of SPI transmission should last 1,25us, so 1 bit should have 0,156 us. It gives us a SPI frequency 6,4 MHz. I decided to use 6Mhz which is very near this value and is in range. My MCU had set 24 MHz so I had to set prescale to 4.

According to table 1 I defined “0” and “1” values in this transmission.

```
#define zero 0b1000000
#define one 0b1111000
```

I also added a ws2812b_color structure to increase efficiency of coding.
I added a buffer including all values transferring to SPI for all leds and reset signal.
ws2812b_color ws2812b_array[WS2812B_LEDS]- array of colors stored for each pixel.

Defined functions:

- void WS2812B_SetDiodeRGB(uint8_t diode_id, uint8_t R, uint8_t G, uint8_t B)
- void WS2812B_SetDiodeHSV(int16_t diode_id, uint16_t Hue, uint8_t Saturation, uint8_t Brightness)
- void WS2812B_CleanBuffer() - setting “0” values in buffer
- void WS2812B_RefreshStrip() - transfer values from ws2812b_array to buffer[] and is used HAL_SPI_Transmit_DMA(&hspi2, buffer, BUFFER_LENGTH) to transmit these value.

Encoder

To read values from encoder I simply set a Timer1 as an encoder and set max value to 80. Which mean that after 360 degrees change its value to 0 because i have 20 step for 360 degrees and at each step it is incremented by 4.
To read this value I simply: encoderValue = htim1.Instance->CNT;

LCD display

To send data to display I used ready library based on i2c in which I had to change slave address.
It has functions like:

- lcd_init() - which initialize transmission
- lcd_send_string(*str) - put string in cursor position
- lcd_put_cur(row,col) - change location of cursor

I added lcd_print_enc_value(encoderValue) - function which helped me to print decimal encoder value in cursor position.

Discrepancies and difficulties

My project description was not much detailed at first. I had a lot freedom manoeuvre so there wasn't any discrepancies.

Problems:

1) At first I had problem with well defining "0" and "1" values for ws2812b in SPI transmission. Problem was in bad clock configuration and I didn't know how buffer works. I changed prescaler to work with 6 MHz SPI.
2) It was hard to send appropriate color for each pixel and use DMA. I increased my buffer to all LED pixels and reset signal in the end, with this it was much easier.
3) Addressing PCF8574AT in I2C. According to datasheet I should have address 0x7C but for some reasons which I don't know it is 0x7D.
4) There was also problem with 5V power supply. I should use external power supply for LEDS but I didn't have. So I had to use 5V from dev board which is of course incorrect.

How it works?

Using button in encoder I switch modes. There are 3 modes: Rainbow, Round pixel and Encoder.

- Rainbow - using HSV I create a rainbow colors at led strip. Changing encoder value we can control speed of rainbow movement.
- Round pixel- there is white pixel which change its position . Changing encoder value we can control speed of pixel movement.
- Encoder- using HSV I set one color at whole led strip. Changing encoder value we can change hue.

These modes and encoder value are printed in LCD display.

Results are shown in this video:
https://www.youtube.com/watch?v=migM74RySeo
it is also available on github:
https://github.com/sebastianwach/Embedded-projects