ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

THIẾT KẾ TRUYỀN THÔNG THEO CHUẨN SPI

Họ và tên: Lý Bảo Khánh
MSV: 21020920

1. NỘI DUNG THÍ NGHIỆM

Thiết lập truyền thông theo chuẩn SPI

Linh kiện sử dụng:

- 2 vi điều khiển STM32F401RET6.

2. NỘI DUNG LÝ THUYẾT

2.1. SPI là gì?

SPI (Serial Peripheral Interface) là chuẩn truyền thông nối tiếp đồng bộ dùng để kết nối và truyền dữ liệu giữa các thiết bị điện tử, được phát triển bởi tập đoàn Motorola. SPI có tốc độ truyền dữ liệu cao, đồng bộ trong việc giao tiếp, cách kết nối đơn giản và tiết kiệm tài nguyên sử dụng. Giao tiếp SPI là 1 giao tiếp thường được dùng trong vi điều khiển với 4 dây. 2 dây để truyền và nhận dữ liệu là MOSI (Master Out Slave In) và MISO (Master In Slave Out). 1 dây SCK để tạo tín hiệu đồng hồ đồng bộ giữa các thiết bị. Còn dây cuối cùng CS để chọn thiết bị sẽ giao tiếp khi trong mạng có nhiều slave.

2.2. Những đặc điểm cơ bản của SPI

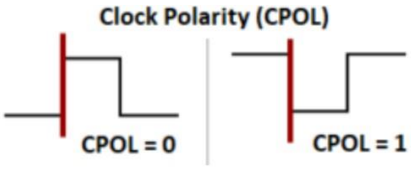
Giao tiếp SPI sử dụng 4 dây để truyền dữ liệu giữa các thiết bị:

- MOSI (Master Out / Slave In) – Đường truyền cho master (thiết bị chủ) gửi dữ liệu đến slave (thiết bị tớ).
- MISO (Master In / Slave Out) – Đường cho slave (thiết bị tớ) gửi dữ liệu đến master (thiết bị chủ).
- SCLK (Serial Clock) hay SCK – Đường truyền của tín hiệu đồng hồ
- CS hay SS (Chip Select, Slave Select): Đường truyền để master chọn slave sẽ gửi dữ liệu đến.

Lợi ích duy nhất của SPI là dữ liệu có thể được truyền mà không bị gián đoạn. Bất kỳ số lượng bit nào cũng có thể được gửi hoặc nhận trong một luồng liên tục.

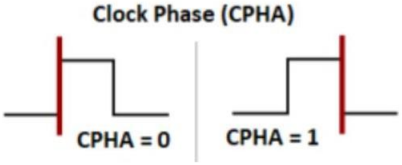
Số dây sử dụng	4
Tốc độ truyền dữ liệu tối đa	10Mbps
Đồng bộ / Không đồng bộ	Đồng bộ
Giao tiếp nối tiếp / song song	Nối tiếp
Masters tối đa	1
Slaves tối đa	Không giới hạn (Lý thuyết)

- *Đồng hồ (SCLK)*
  - Tín hiệu SCK đồng bộ hóa đầu ra của các bit dữ liệu từ master với các slave. Một bit dữ liệu được truyền trong mỗi chu kỳ đồng hồ, do đó tốc độ truyền dữ liệu được xác định bởi tần số của tín hiệu đồng hồ.
  - Bất kỳ giao thức truyền thông nào mà các thiết bị chia sẻ tín hiệu đồng hồ được gọi là đồng bộ. do vậy chuẩn giao tiếp SPI có giao thức truyền thông đồng bộ.
  - Tín hiệu đồng hồ trong SPI có thể đồng bộ hóa bằng cách sử dụng các thuộc tính của phân cực đồng hồ (Clock Polarity) và pha đồng hồ (Clock Phase).
    - Clock Polarity có thể được thiết lập bởi tổng thể để cho phép các bit được xuất ra và lấy mẫu trên cạnh lên hoặc xuống của chu kỳ đồng hồ.



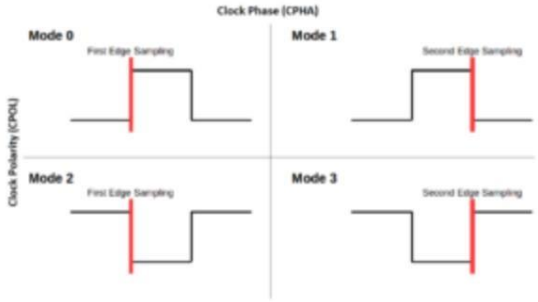
Thuộc tính phân cực đồng hồ Clock Polarity

- Clock Phase có thể được đặt để đầu ra và lấy mẫu xảy ra trên cạnh đầu tiên hoặc cạnh thứ hai của chu kỳ đồng hồ, bất kể nó đang tăng hay giảm.



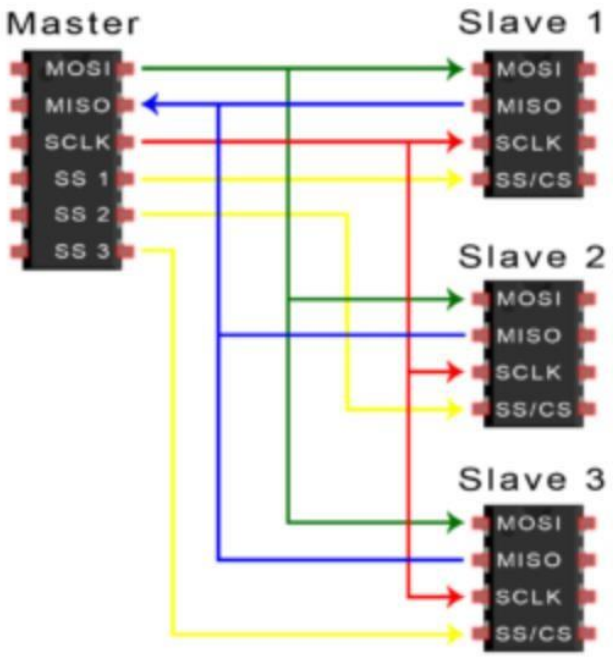
Pha đồng hồ Clock Phase

- Hai thuộc tính này làm việc cùng nhau để xác định khi nào các bit được xuất ra và khi nào chúng được lấy mẫu. 4 chế độ được xác định bởi trạng thái của 2 thuộc tính Clock Polarity và Clock Phase.



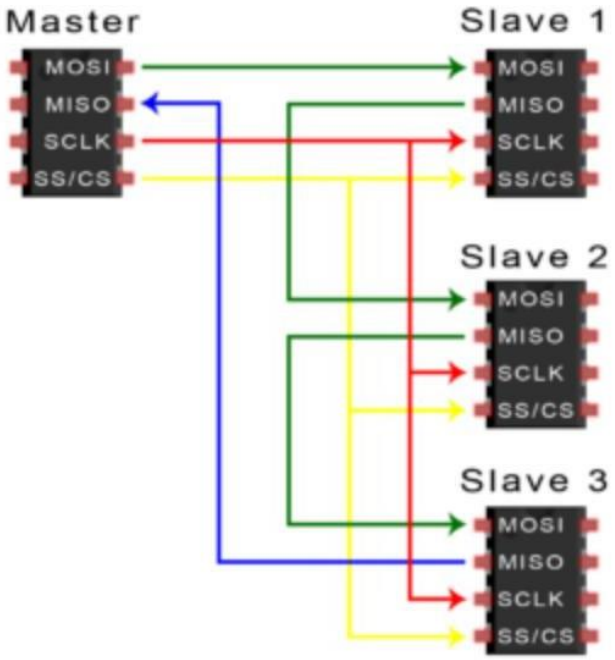
Các chế độ của chuẩn giao tiếp SPI (Click to zoom)

- *Chọn Slave (CS/SS)*
  - Master có thể chọn slave mà nó muốn giao tiếp bằng cách đặt đường dây CS / SS của slave ở mức điện áp thấp (mức 0).
  - Ở trạng thái không giao tiếp, dòng chọn của slave được giữ ở mức điện áp cao (mức 1). Có 2 cách để cấu hình nhiều slave với 1 master tương ứng với số chân CS/SS có sẵn trên master.
    - Nhiều chân CS / SS có thể có sẵn trên master, điều này cho phép đấu dây song song với nhiều slave.



1 Master với nhiều slave trong giao tiếp SPI – Nhiều Chân CS/SS

- Nếu chỉ có một chân CS / SS, nhiều slave có thể được kết nối với master bằng cách nối chuỗi (cấu hình Daisy Chain).



1 Master với nhiều slave trong giao tiếp SPI – 1 Chân CS/SS

- *MOSI và MISO*
    - Master gửi dữ liệu đến slave từng bit, nối tiếp qua đường MOSI. Slave nhận dữ liệu được gửi từ master tại chân MOSI. Dữ liệu được gửi từ master đến slave thường được gửi với bit quan trọng nhất trước, có thể từ bit có trọng số lớn nhất (MSB) hoặc bit có trọng số nhỏ nhất (LSB).
    - Slave cũng có thể gửi dữ liệu trở lại master thông qua đường MISO nối tiếp. Dữ liệu được gửi từ slave trở lại master thường được gửi với bit ít quan trọng nhất trước.

2.3. Cách truyền dữ liệu của chuẩn giao tiếp SPI

Đầu ra chính của tín hiệu đồng hồ

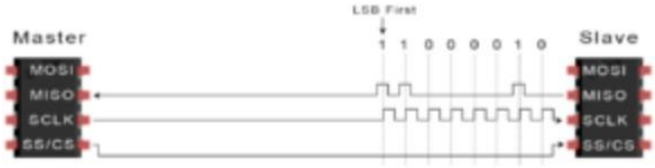
Tín hiệu đồng hồ trong chuẩn giao tiếp SPI

Tín hiệu đồng hồ từ master sẽ cấp cho toàn bộ hệ thống, tạo sự đồng bộ về xung nhịp trong cả master và slave.

- Master chuyển chân SS / CS sang trạng thái điện áp thấp, điều này sẽ kích hoạt chân phụ

Master chọn slave sắp giao tiếp

- Nếu cần phản hồi, Slave sẽ trả lại dữ liệu từng bit một cho Master dọc theo đường MISO. Master đọc các bit khi chúng được nhận



Slave gửi dữ liệu ngược lại qua chân MISO

2.4. Ưu, nhược điểm của chuẩn giao tiếp SPI

a. Ưu điểm

- Không có Start bit và Stop bit như trong giao tiếp I2C và giao tiếp UART. Vì vậy dữ liệu có thể được truyền liên tục mà không bị gián đoạn
- Không có hệ thống định địa chỉ slave phức tạp như I2C
- Tốc độ truyền dữ liệu cao hơn I2C (nhanh gần gấp đôi)
- Các dòng MISO và MOSI riêng biệt, vì vậy dữ liệu có thể được gửi và nhận cùng một lúc

b. Nhược điểm

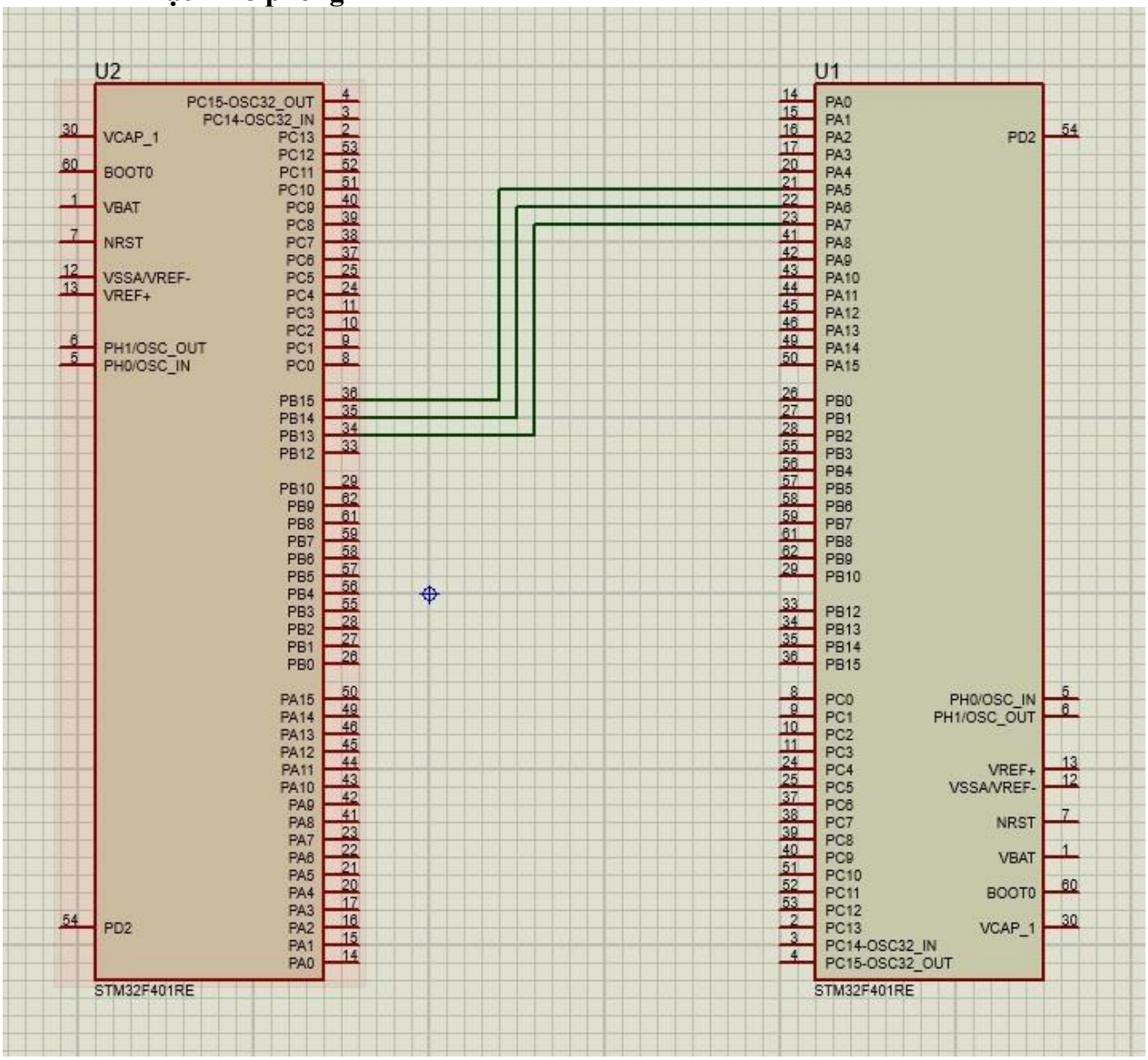
- Sử dụng bốn dây (I2C và UART sử dụng hai dây).
- Không xác nhận rằng dữ liệu đã được nhận thành công.

- Không có hình thức kiểm tra lỗi như bit chẵn lẻ (Parity bit) như trong UART.
- Chỉ cho phép một master duy nhất.

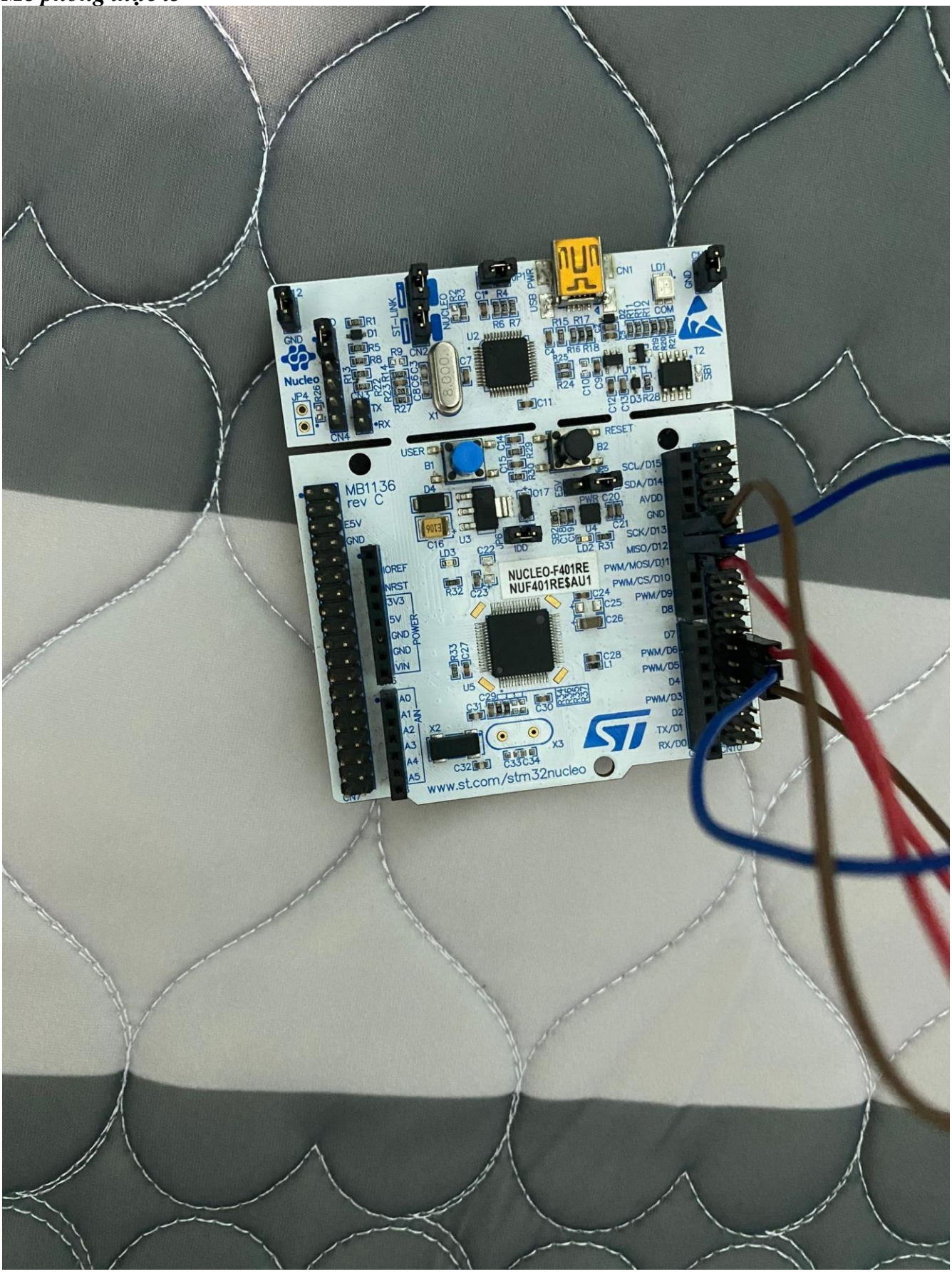
3. NỘI DUNG MÔ PHỎNG

3.1. Mô phỏng trên proteus

- **Mạch mô phỏng**



3.2. Mô phỏng thực tế



Cách mạch hoạt động (do thiếu linh kiện nên ta sẽ dùng dùng 1 vi điều khiển với 2 SPI để giao tiếp):

- Ta cấu hình các chân của vi điều khiển để thực hiện chức năng như một SPI Master và SPI Slave: PA5/PB15, PA6/PB14, PA7/PB13 lần lượt là CLK, MISO và MOSI. Ở đây không cần CS (Slave Select) do chỉ giao tiếp giữa 1 Master và 1 Slave.
- Nối các chân tương ứng của Master và Slave.
- Gửi dữ liệu nhận được ở Slave qua UART đến máy tính để kiểm tra.

3.3. Code.

```
#include "stm32f4xx.h"

//Cấu hình PA5, PA6, PA7 để thực hiện chức năng SPI MASTER
void SPI1_Init(void) {
   RCC->APB2ENR |= RCC_APB2ENR_SPI1EN;
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;

   GPIOA->MODER |= GPIO_MODER_MODE5_1 | GPIO_MODER_MODE6_1;
    GPIOA->MODER |= GPIO_MODER_MODE7_1;
   GPIOA->AFR[0] |= (5 << 20) | (5 << 24) | (5 << 28);

   SPI1->CR1 |= SPI_CR1_BR_0 | SPI_CR1_BR_1;
   SPI1->CR1 |= SPI_CR1_MSTR;
   SPI1->CR1 |= SPI_CR1_SSM | SPI_CR1_SSI;
   SPI1->CR1 |= SPI_CR1_SPE;
}

//Cấu hình PB13, PB14, PB15 để thực hiện chứ năng SPI SLAVE.
void SPI2_Init(void) {
   RCC->APB1ENR |= RCC_APB1ENR_SPI2EN;
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN;

   GPIOB->MODER |= GPIO_MODER_MODE13_1 | GPIO_MODER_MODE14_1;
   GPIOB->MODER |= GPIO_MODER_MODE15_1;
   GPIOB->AFR[1] |= (5 << 20) | (5 << 24) | (5 << 28);

   SPI2->CR1 |= SPI_CR1_BR_0 | SPI_CR1_BR_1;
   SPI2->CR1 &= ~SPI_CR1_MSTR;
   SPI2->CR1 &= ~(SPI_CR1_SSM | SPI_CR1_SSI);
   SPI2->CR1 |= SPI_CR1_SPE;
}

//Gửi dữ liệu qua truyền thông SPI (có thể từ Master tới Slave hoặc ngược lại)
void SPI_Transmit(SPI_TypeDef *SPIx, uint8_t data) {
   while (!(SPIx->SR & SPI_SR_TXE));
   SPIx->DR = data;
   while (SPIx->SR & SPI_SR_BSY);
```

```
}

//Nhận dữ liệu được gửi (từ Slave hoặc Master)
uint8_t SPI_Receive(SPI_TypeDef *SPIx) {
   while (!(SPIx->SR & SPI_SR_RXNE));
   return SPIx->DR;
}

//Cấu hình COM ảo gửi dữ liệu lên máy tính để kiểm tra
void UART2_Init(void) {
   RCC->APB1ENR |= RCC_APB1ENR_USART2EN;
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;

   GPIOA->MODER |= GPIO_MODER_MODE2_1 | GPIO_MODER_MODE3_1;
   GPIOA->AFR[0] |= 0x7700;

   USART2->BRR = 0x683;
   USART2->CR1 |= USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
}

//Gửi dữ liệu để kiểm tra
void UART2_Send(uint8_t data) {
   while (!(USART2->SR & USART_SR_TXE));
   USART2->DR = data;
}

//Thực hiện truyền thông SPI rồi kiểm tra trên máy tính.
int main(void) {
   SPI1_Init();
   SPI2_Init();
            UART2_Init();
   uint8_t tx_data = 'h';
   uint8_t rx_data;

   while (1) {
                        SPI_Transmit(SPI1,  tx_data);
                        rx_data = SPI_Receive(SPI2);
                        UART2_Send(rx_data);
                        for (int i = 0 ; i < 10000000; i++);
   }
}
```