ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

THIẾT KẾ TRUYỀN THÔNG THEO CHUẨN UART

Họ và tên: Lý Bảo Khánh
MSV: 21020920

1. YÊU CẦU THỰC NGHIỆM

Thiết lập truyền thông nối tiếp theo chuẩn UART

Linh kiện sử dụng:

- STM32F103C8T6 (Receiver)
- STM32F401RET6 (Transmitter)
- CP2102

2. NỘI DUNG LÝ THUYẾT TRUYỀN THÔNG NỐI TIẾP THEO CHUẨN RS232

2.1. RS232 là gì?

RS232 (Recommended Standard 232) là một chuẩn truyền thông nối tiếp được phát triển bởi Hiệp hội Công nghiệp Điện tử (EIA) và Hiệp hội Công nghiệp Viễn thông (TIA).

Chuẩn này đề cập đến việc tuyền dữ liệu nối tiếp giữa một thiết bị chủ (DTE - Data Terminal Equipment) và một thiết bị ngoại vi (DCE - Data Circuit-Terminating Equipment).

2.2. Những đặc tính của chuẩn RS232

Chiều dài cực đại	15m (50 Feet)
Tốc độ dữ liệu cực đại	20Kps
Điện áp ngõ ra cực đại	±25V
Điện áp ngõ ra có tải	±5V đến ±15V
Trở kháng tải	3K đến 7K
Điện áp ngõ vào	±15V
Độ nhạy ngõ vào	±3V
Trở kháng ngõ vào	3K đến 7K

Mức logic của RS232 được định nghĩa khác với logic TTL. Ở ngõ ra của mạch lái, mức cao (tương ứng với logic 0) là một điện áp từ +5 đến +15V, còn mức thấp (tương ứng với logic 1) là một điện áp từ -5V đến - 15V. Tại ngõ vào của một bộ thu, mức cao được định nghĩa từ +3V đến +15V (gọi là space), mức thấp được định nghĩa từ -3V đến -15V (gọi là mark)

Để giảm nguy cơ bị nhiễu giữa các tín hiệu kế cận, tốc độ thay đổi (slew rate) được giới hạn tối đa là 30V/µs, và tốc độ được giới hạn là 20kbps.

Trở kháng nhìn bởi mạch lái được định nghĩa là từ 3 đến 7 kΩ. Tải dung tối đa của đường truyền cũng được giới hạn là 2500 pF, và như vậy tùy thuộc vào loại cáp mà chiều dài tối đa có thể được xác định từ điện dung trên đơn vị chiều dài của cáp.

2.3. Truyền dữ liệu không đồng bộ và đồng bộ

a. Truyền thông dữ liệu không đồng bộ

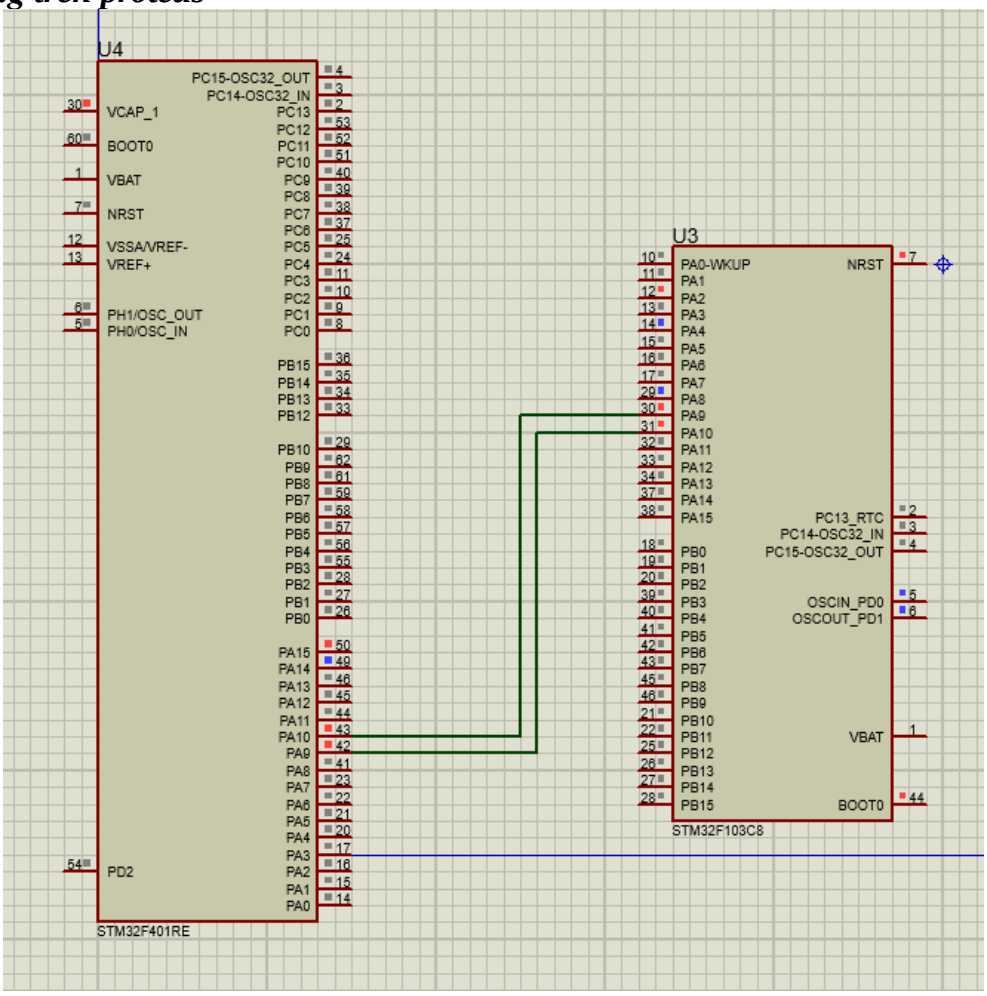
Các bit dữ liệu không được đồng bộ hoá bởi xung đồng hồ. Truyền dữ liệu không đồng bộ thường sử dụng trong các ứng dụng như modem và máy tính

b. Truyền thông dữ liệu đồng bộ

Các bit dữ liệu được đồng hoá bởi xung đồng hồ. Truyền dữ liệu đồng bộ thường được sử dụng trong các ứng dụng yêu cầu độ chính xác cao như truyền thông giữa máy tính và máy in

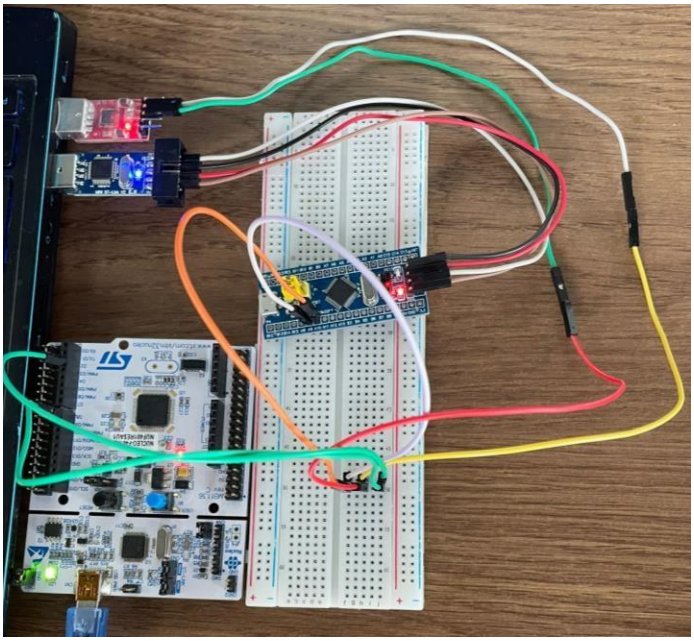
3. NỘI DUNG THỰC NGHIỆM

3.1. Mô phỏng trên proteus



3.2. Thực nghiệm thực tế

- Ở đây ta nối chéo chân Tx và Rx của 2 chân điều khiển (Tx nối Rx), riêng với Receiver ta nối chân Tx qua CP2102 đến cổng COM của máy tính để kiểm tra sự truyền dữ liệu.
- Do sự truyền dữ liệu ở đây thực chất là sự truyền điện áp rồi giải mã thành các ký tự nên khi dây nối Rx và Tx quá xa thì dữ liệu có thể bị sai lệch.
- Trong quá trình thực nghiệm thì có thể thấy rõ sự ảnh hưởng này: khi truyền ký tự '1', mức điện áp ở chân Tx của Transmitter ở khoảng 1.26328V, nhưng khi cắm qua board tới chân Rx của Receiver thì mức điện áp chỉ còn khoảng 1.2117V, khiến cho ký tự nhận được là '/'.
- Lưu ý cần cắm sát chân nhất có thể hoặc có thể cắm thẳng vào đầu nối của các chân Rx và Tx để tránh sự mất mát điện áp, dẫn đến sự sai lệch trong dữ liệu nhận được và dữ liệu nhận được.



3.3. Code

3.3.1. Code trên STM32F103C8T6 (Receiver)

```
#include "stm32f10x.h"

//Gửi ký tự nhận được lên máy tính để kiểm tra
void USART1_Transmit(char c)
{
  while (!(USART1->SR &
  USART_SR_TXE)); USART1->DR =
  c;
}

//Xử lý ngắt khi nhận được tín hiệu từ Transmitter
void USART1_IRQHandler(void)
{
  if (USART1->SR & USART_SR_RXNE)
  {
      char c = USART1-
      >DR; USART2-
      >DR = c;
  }
}

//Cấu hình cho chân UART1 (ở đây là PA9 và PA10)
void USART1_Init(void)
{
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_USART1EN;
```

```
  GPIOA->CRH |= GPIO_CRH_MODE9 | GPIO_CRH_CNF9_1 |
  GPIO_CRH_CNF10_0; GPIOA->CRH &= ~(GPIO_CRH_MODE10 |
  GPIO_CRH_CNF9_0 | GPIO_CRH_CNF10_1);

  USART1->BRR = 0x341;
  USART1->CR1 |= USART_CR1_UE;
  USART1->CR1 |= USART_CR1_TE |
  USART_CR1_RE;
}


int main(void)
{
  USART1_Init();
  while (1)
  {
  }
  return 0;
}
```

3.3.2. *Code trên STM32F401RET6 (Transmitter)*

```
#include "stm32f4xx.h"

//Cấu hình cho UART
void UART_Init(void) {
  RCC->AHB1ENR |=
  RCC_AHB1ENR_GPIOAEN; RCC-
  >APB2ENR |=
  RCC_APB2ENR_USART1EN;

  GPIOA->MODER |= GPIO_MODER_MODER9_1 |

  GPIO_MODER_MODER10_1; GPIOA->AFR[1] |= 0x00000770;

  USART1->BRR = 0x683;

  USART1->CR1 |= USART_CR1_UE;

  USART1->CR1 |= USART_CR1_TE | USART_CR1_RE;

  USART1->CR1 |=
  USART_CR1_RXNEIE;
  NVIC_EnableIRQ(USART1_IRQn);
}
//Hàm truyền ký tự sang Receiver
void UART_Transmit(char c) {
  while (!(USART1->SR &
  USART_SR_TXE)); USART1->DR =
```

```
  c;
}
//Truyền chuỗi ký tự sang Receiver
void UART_Transmit_String(char*
  str) { while (*str) {
    UART_Transmit(*str++);
  }
}
int
  main(void)
  {
  UART_Init
  ();     while
  (1) {
    UART_Transmit_String("Hello World\n");
                                for (int i =0; i< 9000000;i++);
  }
  return 0;
}
```