

BÁO CÁO THỰC HÀNH BÀI 3

Môn học: CHUYÊN ĐỀ THIẾT KẾ HỆ THỐNG NHÚNG 1- Mã lớp: CE437.N11

Giảng viên hướng dẫn thực hành: Phạm Minh Quân

Thông tin các sinh viên	Mã số sinh viên	Họ và tên sinh viên
	19520887	Phạm Trung Quốc
	19521651	Phạm Trọng Huỳnh
	19520928	Viên Minh Tân
	19520036	Phạm Quốc Đăng
Link các tài liệu tham khảo (nếu có)		
Đánh giá của giảng viên: + Nhận xét + Các lỗi trong chương trình + Gợi ý		

MỤC LỤC

1. Nội dung thực hành	3
2. Thiết lập các cấu hình liên quan	3
3. Mã nguồn và kết quả.....	5
a. Sơ đồ giải thuật	5
b. Mã nguồn	6
c. Kết quả	8

1. Nội dung thực hành

Sinh viên lập trình giao tiếp giữa 2 module CAN1 và CAN2



CAN1 là node 1: (TX_ID: 0x012 ; RX_ID: 0x0A2)

CAN2 là node 2: (TX_ID: 0x0A2 ; RX_ID: 0x012)

Node 2 gửi gói tin gồm: byte 0 đến byte 6 đều có giá trị: 0xAA và byte còn lại chứa giá trị đếm số thứ tự của gói tin được gửi. (0x00-0xFF) đến Node 1. Node 1 nhận gói tin từ Node 2 và tăng giá trị của byte 7 lên 1 đơn vị. Sau đó gửi lại gói tin cho Node 2 gồm: byte 0 đến byte 5 có giá trị: 0x55. Byte 6 là giá trị nhận được từ byte 7 của gói tin nhận được tăng lên 1. Byte 7 là checksum của các dữ liệu byte0-byte6. Check sum được tính theo chuẩn CRC 8 SAE J1850. Chu kỳ gửi gói tin của CAN1 là 200ms và chu kỳ gửi gói tin của CAN2 là 500 ms. Sinh viên sử dụng cổng UART để hiển thị gói tin và các LEDs để hiển thị trạng thái kiểm tra checksum sau khi nhận được gói tin.

2. Thiết lập các cấu hình liên quan

Cấu hình sử dụng Xung từ thạch anh ngoại. Khi sử dụng thạch anh ngoại giúp tốc độ vi điều khiển đạt tối đa được khả năng của nó. Với HSE là nguồn xung chính để nuôi MCU và các ngoại vi, thường sử dụng thạch anh là 8Mhz (giá trị này sẽ được cấu hình ở tab Clock Configuration). LSE là nguồn xung cung cấp cho bộ RTC hoạt động và sử dụng cho việc backup khi nguồn VDD bị mất.

High Speed Clock (HSE) 
Low Speed Clock (LSE) 

Cấu hình TIMER 2 và TIMER 3. Vì hai TIMER này chạy độc lập và song song với chương trình hệ thống. Với TIMER2 chu kỳ tràn là 200ms và TIMER3 chu kỳ tràn là 500ms. Sử dụng công thức sau để tính được chu kỳ tràn của 1 TIMER:

$$\frac{\text{Clock system}}{(\text{Prescaler}+1)(\text{Counter Period}+1)} = \frac{1}{\text{Over period}}$$

▼ Counter Settings
Prescaler (PSC - 16 bits val... 8399
Counter Mode Up
Counter Period (AutoReload... 1999

▼ Counter Settings

Prescaler (PSC - 16 bits val... 8399

Counter Mode Up

Counter Period (AutoReload... 4999

Cấu hình cho CAN 1 và CAN 2 tương tự nhau với các thông số như sau

▼ Bit Timings Parameters

Prescaler (for Time Quantum) 4

Time Quantum 95.23809523809524 ns

Time Quanta in Bit Segmen... 16 Times

Time Quanta in Bit Segmen... 4 Times

Time for one Bit 2000 ns

Baud Rate 500000 bit/s

ReSynchronization Jump Wi... 2 Times

▼ Basic Parameters

Time Triggered Communica... Disable

Automatic Bus-Off Manage... Enable

Automatic Wake-Up Mode Enable

Automatic Retransmission Disable

Receive Fifo Locked Mode Disable

Transmit Fifo Priority Disable

Cấu hình USART1 phục vụ cho mục đích nạp code, bật chế độ “Asynchronous mode”. Với chân 2 chân TX và RX lần lượt là PA9 và PA10

Mode

Hardware Flow Control (RS232)

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum
PA9	USART1_TX	n/a	Alternate F...	No pull-up ...	Very High
PA10	USART1_RX	n/a	Alternate F...	No pull-up ...	Very High

Cấu hình xung tại tab Clock Configuration. Tại đây chúng ta cấu hình giá trị xung ngoại cũng như luồng đi để hệ thống tính toán xung cho hệ thống thông qua các bộ chia và bộ nhân.

a. Sơ đồ giải thuật

Mỗi khi hàm HAL_CAN_AddTxMessage được gọi gói tin được nằm sẵn trong Tx của node gửi sẽ được thêm vào buffer. Sau khi hàm này được gọi, hàm HAL_CAN_RxFifo0MsgPendingCallback được kích hoạt với chức năng callback, ở đây gọi hàm HAL_CAN_GetRxMessage để lấy gói tin từ buffer về Rx của node nhận.

b. Mã nguồn

Bằng việc cấu hình TIMER 2 và TIMER 3 như trên, sau mỗi lần timer tràn hàm HAL_TIM_PeriodElapsedCallback sẽ được gọi để thực hiện chức năng của một callback mà chúng ta đã thiết lập để bật cờ cho phép Node 2, Node 1 gửi các gói tin như theo yêu cầu bài toán. Với biến Node2to1 = 1 cho phép Node 2 gửi gói tin đến Node 1, tương tự với biến Node1to2. Với việc sử dụng 2 hàm if trong hàm callback này sẽ đảm bảo được mỗi khi có 1 timer tràn theo đúng chu kỳ của nó thì chương trình sẽ thực hiện việc gửi các gói tin.

```
119 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
120 {
121     if(htim->Instance == htim2.Instance)
122     {
123         // HAL_GPIO_TogglePin(GPIOB, LED1_Pin);
124         // printf("gud timer 2\r\n");
125         Node1to2 = 1;
126     }
127     if(htim->Instance == htim3.Instance)
128     {
129         // HAL_GPIO_TogglePin(GPIOB, LED2_Pin);
130         // printf("gud timer 3\r\n");
131         Node2to1 = 1;
132     }
133 }
```

Trước khi gửi cần chuẩn bị gói tin. Đối với Node 1 với ID là 0x012 và Node 2 là 0x0A2. Node 2 gửi gói tin gồm: byte 0 đến byte 6 đều có giá trị: 0xAA và byte còn lại chứa giá trị đếm số thứ tự của gói tin được gửi. (0x00-0xFF) đến Node 1. Node 1 nhận gói tin từ Node 2 và tăng giá trị của byte 7 lên 1 đơn vị. Sau đó gửi lại gói tin cho Node 2 gồm: byte 0 đến byte 5 có giá trị: 0x55. Byte 6 là giá trị nhận được từ byte 7 của gói tin nhận được tăng lên 1. Byte 7 là checksum của các dữ liệu byte0-byte6. Phần code này sẽ được đặt trong main.

```
Node1_TxHeader.DLC = 8; // data length      Node2_TxHeader.DLC = 8; // data length
Node1_TxHeader.IDE = CAN_ID_STD;           Node2_TxHeader.IDE = CAN_ID_STD;
Node1_TxHeader.RTR = CAN_RTR_DATA;         Node2_TxHeader.RTR = CAN_RTR_DATA;
Node1_TxHeader.StdId = 0x012; // ID         Node2_TxHeader.StdId = 0x0A2; // ID

Node1_TxData[0] = 0x55;                    Node2_TxData[0] = 0xAA;
Node1_TxData[1] = 0x55;                    Node2_TxData[1] = 0xAA;
Node1_TxData[2] = 0x55;                    Node2_TxData[2] = 0xAA;
Node1_TxData[3] = 0x55;                    Node2_TxData[3] = 0xAA;
Node1_TxData[4] = 0x55;                    Node2_TxData[4] = 0xAA;
Node1_TxData[5] = 0x55;                    Node2_TxData[5] = 0xAA;
Node1_TxData[6] = 0;                      Node2_TxData[6] = 0xAA;
Node1_TxData[7] = 0;                      Node2_TxData[7] = 0x0;
```

Với phần code sau được đặt trong vòng lặp while, có nghĩa hàm if này sẽ luôn được kiểm tra để biết rằng liệu Node 1 hay Node 2 đã đến lúc gửi các gói tin của mình hay chưa, thời điểm gửi sẽ phụ thuộc vào timer 2 và 3 như đã thiết lập bên trên. Hàm printf được gọi để check gói tin trước khi gửi đi. Hàm HAL_CAN_AddTxMessage được gọi để tiến hành thêm gói tin vào buffer từ Node Tx để tiến hành gọi đến callback. Sau đó cờ Node2to1 hoặc Node1to2 sẽ được trả về 0 để chờ lần gửi tiếp theo.

```
if (Node2to1 == 1)
{
    Node2_TxData[7] ++; // so gói tin đã gửi đi
    printf("N2 sent %x.%x.%x.%x.%x.%x.%x.%x ", Node2_TxData[0],Node2_TxData[1],Node2_TxData[2],Node2_TxData[3], Node2_TxData[4],
    printf("Header of Tx N2 %x.%x.%x.%x\r\n",Node2_TxHeader.StdId,Node2_TxHeader.RTR, Node2_TxHeader.IDE, Node2_TxHeader.DLC);
    HAL_CAN_AddTxMessage(&hcan2, &Node2_TxHeader, Node2_TxData,
        &Node2_TxMailbox);
    Node2to1 = 0; // flag for timer
}
if (Node1to2 == 1)
{
    Node1_TxData[6] = Node2_TxData[7] + 1;
    Node1_TxData[7] = calc_SAE_J1850(Node1_TxData, 7);
    printf("N1 sent %x.%x.%x.%x.%x.%x.%x.%x ", Node1_TxData[0],Node1_TxData[1],Node1_TxData[2],Node1_TxData[3], Node1_TxData[4],
    printf("Header of Tx N1 %x.%x.%x.%x\r\n",Node1_TxHeader.StdId,Node1_TxHeader.RTR, Node1_TxHeader.IDE, Node1_TxHeader.DLC);
    HAL_CAN_AddTxMessage(&hcan1, &Node1_TxHeader, Node1_TxData,
        &Node1_TxMailbox);
    Node1to2 = 0; // flag for timer
}
```

Sau khi được hàm HAL_CAN_AddTxMessage được gọi sẽ kích hoạt hàm HAL_CAN_RxFifo0MsgPendingCallback. Hàm HAL_CAN_GetRxMessage được gọi để lấy gói tin từ Buffer về Node nhận. Ở đây có hàm printf để kiểm tra gói tin sau khi nhận được có giống như gói tin đã gửi từ Node gửi hay không.

```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
    HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &Node1_RxHeader, Node1_RxData);
    if (Node1_RxHeader.DLC == 8)
    {
        printf("N1 get %x.%x.%x.%x.%x.%x.%x.%x ", Node1_RxData[0],Node1_RxData[1],Node1_RxData[2],Node1_RxData[3], Node1_RxData[4],
        printf("Header of Rx N1 %x.%x.%x.%x\r\n",Node1_RxHeader.StdId,Node1_RxHeader.RTR, Node1_RxHeader.IDE, Node1_RxHeader.DLC);
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
    }
}

void HAL_CAN_RxFifo1MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
    HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO1, &Node2_RxHeader, Node2_RxData);
    if (Node2_RxHeader.DLC == 8)
    {
        printf("N2 get %x.%x.%x.%x.%x.%x.%x.%x ", Node2_RxData[0],Node2_RxData[1],Node2_RxData[2],Node2_RxData[3], Node2_RxData[4],
        printf("Header of Rx N2 %x.%x.%x.%x\r\n",Node2_RxHeader.StdId,Node2_RxHeader.RTR, Node2_RxHeader.IDE, Node2_RxHeader.DLC);
        HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
    }
}
```

c. Kết quả

Δ Kiểm tra kết quả bằng Terminal

```
14:48:04.899 -> begin program ...
14:48:04.899 -> N2 sent aa.aa.aa.aa.aa.aa.1 Header of Tx N2 a2.0.0.8
14:48:04.899 -> N1 sent 55.55.55.55.55.55.2.de Header of Tx N1 12.0.0.8
14:48:05.098 -> N1 sent 55.55.55.55.55.55.2.de Header of Tx N1 12.0.0.8
14:48:05.297 -> N1 sent 55.55.55.55.55.55.2.de Header of Tx N1 12.0.0.8
14:48:05.397 -> N2 sent aa.aa.aa.aa.aa.aa.2 Header of Tx N2 a2.0.0.8
14:48:05.496 -> N1 sent 55.55.55.55.55.55.3.cf Header of Tx N1 12.0.0.8
14:48:05.695 -> N1 sent 55.55.55.55.55.55.3.cf Header of Tx N1 12.0.0.8
14:48:05.894 -> N1 sent 55.55.55.55.55.55.3.cf Header of Tx N1 12.0.0.8
14:48:05.894 -> N2 sent aa.aa.aa.aa.aa.aa.3 Header of Tx N2 a2.0.0.8
14:48:06.092 -> N1 sent 55.55.55.55.55.55.4.b8 Header of Tx N1 12.0.0.8
14:48:06.291 -> N1 sent 55.55.55.55.55.55.4.b8 Header of Tx N1 12.0.0.8
14:48:06.391 -> N2 sent aa.aa.aa.aa.aa.aa.4 Header of Tx N2 a2.0.0.8
14:48:06.490 -> N1 sent 55.55.55.55.55.55.5.a9 Header of Tx N1 12.0.0.8
14:48:06.689 -> N1 sent 55.55.55.55.55.55.5.a9 Header of Tx N1 12.0.0.8
14:48:06.888 -> N1 sent 55.55.55.55.55.55.5.a9 Header of Tx N1 12.0.0.8
14:48:06.888 -> N2 sent aa.aa.aa.aa.aa.aa.5 Header of Tx N2 a2.0.0.8
14:48:07.087 -> N1 sent 55.55.55.55.55.55.6.9a Header of Tx N1 12.0.0.8
14:48:07.286 -> N1 sent 55.55.55.55.55.55.6.9a Header of Tx N1 12.0.0.8
14:48:07.386 -> N2 sent aa.aa.aa.aa.aa.aa.6 Header of Tx N2 a2.0.0.8

14:45:20.847 -> begin program ...
14:45:20.847 -> N1 get aa.aa.aa.aa.aa.aa.1 Header of Rx N1 a2.0.0.8
14:45:20.880 -> N2 get [55.55.55.55.55.55.2.de Header of Rx N2 12.0.0.8
14:45:21.046 -> N2 get [55.55.55.55.55.55.2.de Header of Rx N2 12.0.0.8
14:45:21.245 -> N2 get [55.55.55.55.55.55.2.de Header of Rx N2 12.0.0.8
14:45:21.344 -> N1 get aa.aa.aa.aa.aa.aa.2 Header of Rx N1 a2.0.0.8
14:45:21.443 -> N2 get [55.55.55.55.55.55.3.cf Header of Rx N2 12.0.0.8
14:45:21.642 -> N2 get [55.55.55.55.55.55.3.cf Header of Rx N2 12.0.0.8
14:45:21.841 -> N1 get aa.aa.aa.aa.aa.aa.3 Header of Rx N1 a2.0.0.8
14:45:21.874 -> N2 get [55.55.55.55.55.55.4.b8 Header of Rx N2 12.0.0.8
14:45:22.073 -> N2 get [55.55.55.55.55.55.4.b8 Header of Rx N2 12.0.0.8
14:45:22.271 -> N2 get [55.55.55.55.55.55.4.b8 Header of Rx N2 12.0.0.8
14:45:22.371 -> N1 get aa.aa.aa.aa.aa.aa.4 Header of Rx N1 a2.0.0.8
14:45:22.470 -> N2 get [55.55.55.55.55.55.5.a9 Header of Rx N2 12.0.0.8
14:45:22.669 -> N2 get [55.55.55.55.55.55.5.a9 Header of Rx N2 12.0.0.8
14:45:22.868 -> N1 get aa.aa.aa.aa.aa.aa.5 Header of Rx N1 a2.0.0.8
14:45:22.868 -> N2 get [55.55.55.55.55.55.6.9a Header of Rx N2 12.0.0.8
14:45:23.067 -> N2 get [55.55.55.55.55.55.6.9a Header of Rx N2 12.0.0.8
14:45:23.266 -> N2 get [55.55.55.55.55.55.6.9a Header of Rx N2 12.0.0.8
14:45:23.365 -> N1 get aa.aa.aa.aa.aa.aa.6 Header of Rx N1 a2.0.0.8
```

Kiểm tra kết quả bằng terminal, thấy được quá trình gửi gói tin theo chu kỳ của các node đúng với yêu cầu bài toán (Node 2 gửi đến Node 1 theo chu kỳ 500ms, Node 1 gửi đến Node 2 theo chu kỳ 200ms). Bên cạnh đó, các yêu cầu như Node1_RxData[7] phụ trách nhiệm vụ đếm số gói tin nhận được từ Node2, Node2_RxData[6] = Node1_TxData[7] + 1 và Node2_Rxdata[7] là checksum của bit 0 đến bit 6 của gói tin nhận được từ Node 1, cũng được thực hiện đúng đắn.

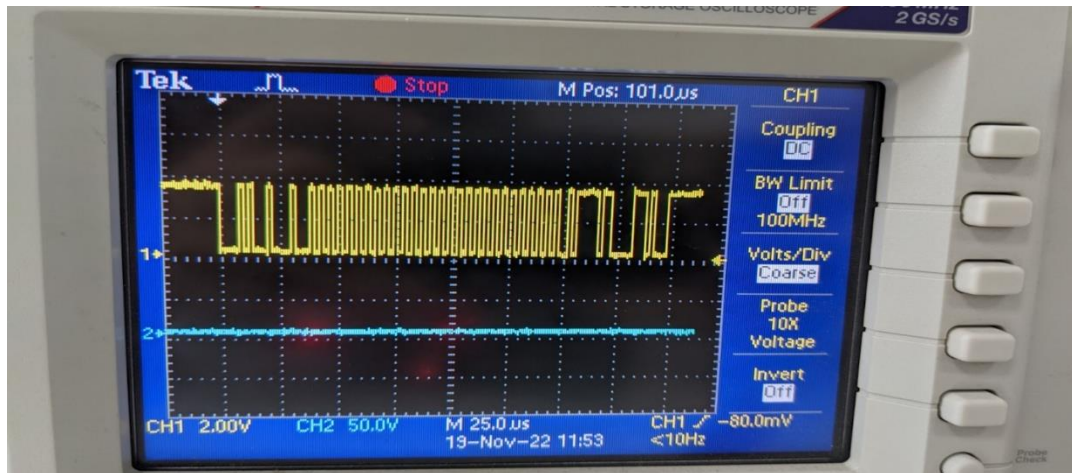
Kiểm tra phần Header của các node bao gồm ID, RTR, IDE, DLC đã nhận đúng như mong đợi. Với ID lần lượt (node 1: (TX_ID: 0x012; RX_ID: 0x0A2) node 2: (TX_ID: 0x0A2; RX_ID: 0x012)), với RTR là bit 0 (Data Frame), IDE là bit 0 standard frame (phân biệt standard frame và extended frame), với DLC là số bytes data (ở đây mang giá trị có 8 bytes dữ liệu).

Ban đầu nhóm thực hiện xuất 2 thông tin gói tin trước và sau khi nhận để so sánh ra cùng một thời điểm terminal để tiện cho việc debug và quan sát tính đúng đắn của gói tin, nhưng đã bị

lỗi phần printf, in quá nhiều thông tin tại một thời điểm đã khiến dữ liệu in ra có phần không thuận mắt người xem, nên nhóm đã quyết định tác ra tiện cho việc ghi báo cáo.

Δ Kiểm tra kết quả bằng bộ dao động ký

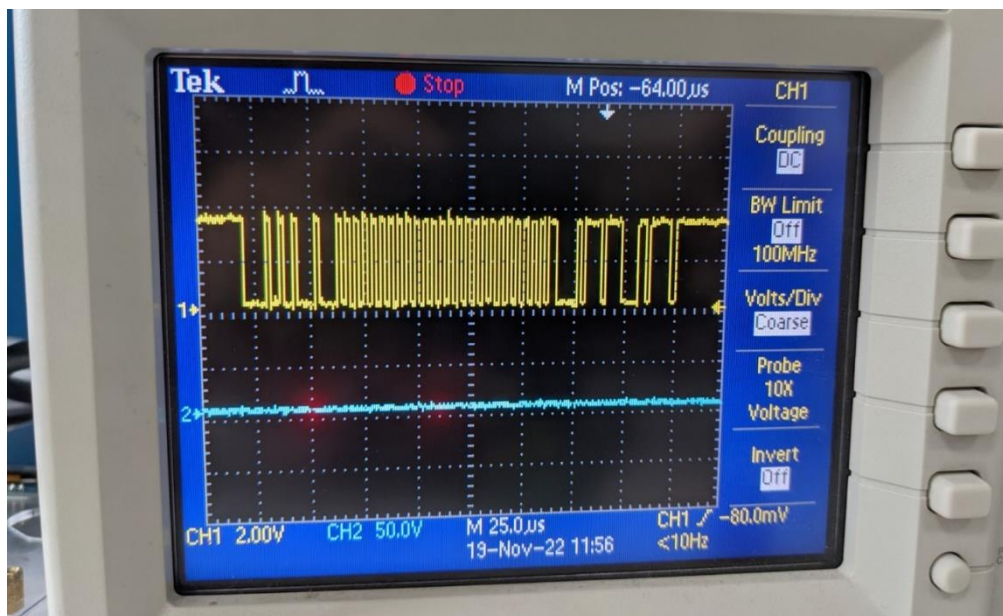
Tiến hành kiểm tra gói tin trước và sau khi gửi ở cả 2 node. Đầu tiên là việc gửi từ node 2 qua node 1, và gói tin đầu tiên đó chính là gói tin trước khi gửi đi, nằm trong biến Node2_Tx.



Gói tin trên được viết lại như sau:

```
0000.1010.0010 | 001000 | 0101.0101.0101.0101. 0101.0101.0101.0010 | 1111.0110.0001.1101.000
[ 0x0A2 ][ Ctrl ][ 7 bytes dữ liệu mang giá trị 0x55 ][count][ phần còn lại của gói tin ]
```

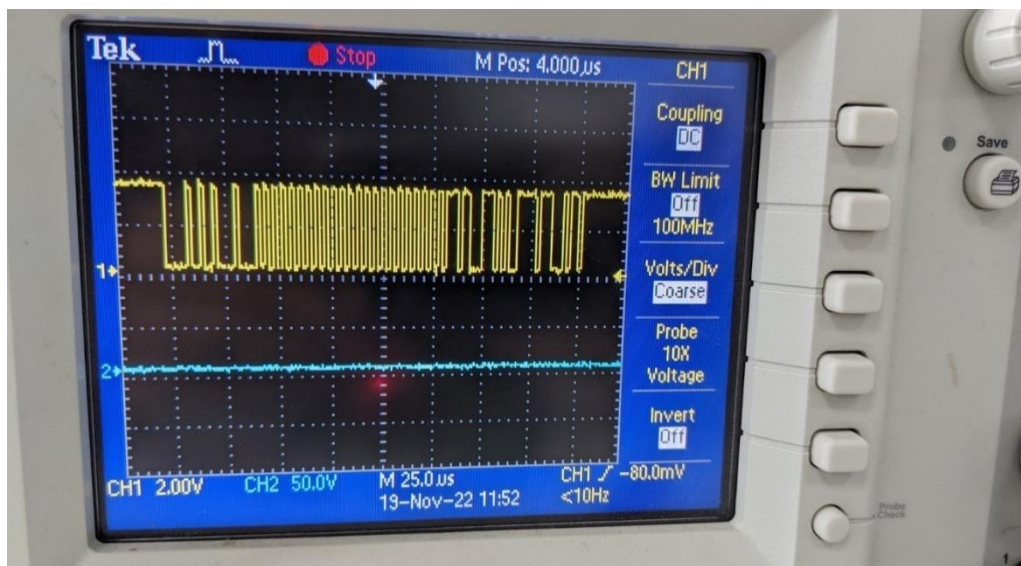
Kiểm tra gói tin mà Node1_Rx nhận được từ Node2_Tx.



Kiểm tra gói tin Node1_Tx trước khi gửi đi gói tin đến Node2



Kiểm tra gói tin Node2_Rx nhận được từ Node 1



Ở những lần đo bằng máy đo dao động, các tín hiệu nhận được không như mong đợi và nó giống với những gì đã hiển thị ở terminal. Đến nay nhóm vẫn chưa tìm ra nguyên nhân lỗi cũng như cách khắc phục nó.