UNIVERSIDADE FEDERAL DE ITAJUBÁ
Instituto de Engenharia de Sistemas e Tecnologias da Informação
Equipe Cheetah E-Racing

Teste do protocolo de comunicação CAN

Caio Tácito Borges da Costa
Itajubá - Minas Gerais
03 de Maio de 2021

Resumo

A implementação do protocolo CAN no sistema de telemetria do CE-21 depende da interoperabilidade entre todos os controladores CAN do veículo. Portanto, é necessário garantir que o hardware e software desenvolvidos pela equipe são capazes de se comunicar de forma confiável, econômica e escalável.

1 Objetivos

Serão realizados 4 testes com os componentes para validar cada uma das situações abaixo:

1. Comunicação entre um microcontrolador sem periférico CAN (ESP32 e ATmega328p) e o controlador externo (MCP2515);
2. Comunicação entre um controlador CAN externo e o barramento;
3. Comunicação entre um microcontrolador com periférico CAN (STM32) e o barramento através do transceiver TJA1050;
4. Comunicação entre dois microcontroladores STM32 através do TJA1050;

2 Equipamentos

Para os testes foram utilizados:

- 1 Protoboard;
- 2 Placas de desenvolvimento *blue pill* com microcontroladores STM32F103C8T6 (64k);
- 1 Programador STLink v2;
- 1 Placa de desenvolvimento Arduino com microcontrolador ATmega328p;
- 1 Placa de desenvolvimento Arduino com microcontrolador ATmega2560;
- 2 Módulos CAN MCP2515 + TJA1050;
- 2 Módulos TJA1050 com resistor de terminação de 120 Ω;
- 2 Optoacopladores de alta velocidade 6N137;
- Componentes passivos e jumpers de protoboard.

3 Procedimentos

3.1 Teste MCP2515 + Arduino

O primeiro teste consiste em configurar os registradores do controlador MCP2515 e prepará-lo para fazer parte de um barramento CAN.

3.1.1 Registradores

Os seguintes registradores foram configurados, na ordem:

Nome	Endereço	HEX	BIN
CANINTE	0x2B	0x03	00000011
BFPCTRL	0x0C	0x3C	00111100
TXRTSCTRL	0x0D	0x00	00000000
RXB0CTRL	0x60	0x64	01100100
RXB0CTRL	0x70	0x60	01100000

3.1.2 Configuração Serial

Para visualizar o resultado, utilizou-se a porta UART da placa Arduino UNO configurada a uma velocidade de 115200 bps;

3.1.3 Circuito

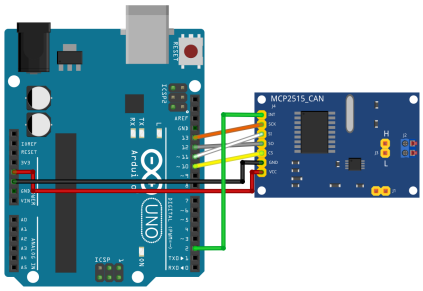

Figura 1: Ligação do módulo CAN nos pinos SPI

3.1.4 Resultado esperado

O controlador CAN deve inicializar corretamente e reportar o resultado *CAN OK* pela interface Serial. A biblioteca utilizada retorna o resultado da inicialização como um valor inteiro, e seus possíveis estados estão descritos abaixo:

Resultado	Valor
CAN OK	0
CAN FAILINIT	1
CAN FAILTX	2
CAN NOMSG	3
CAN CTRLERROR	4
CAN GETTXBFTIMEOUT	5
CAN SENDMSGTIMEOUT	6
CAN FAIL	255

3.2 Teste MCP2515 Bidirecional

O segundo teste verifica a capacidade de envio e recebimento de frames CAN com dois módulos idênticos

ligados no barramento. As configurações dos registradores e periférico serial permanecem as mesmas do teste anterior.

3.2.1 Frame CAN

Será enviado um único frame CAN a uma frequência de 1000Hz. Os 4 primeiros bytes de dados são valores hexadecimais fixos, enquanto os últimos 4 campos variam com o passar do tempo:

ID	MSG1	MSG2	MSG3	MSG4	MSG5	MSG6	MSG7	MSG8
0x01	0xFC	0xFE	0xAB	0xAC	millis()%255	random(0,255)	random(0,16)	random(0,1)

3.2.2 Recebimento de dados

Os dados recebidos pela porta serial do computador serão validados e contabilizados por uma versão modificada do script principal do sistema supervisório da equipe. O programa rodará por 5 minutos calculados através do comando "Date.now()" (milissegundos após o epoch) e finalizará automaticamente após o tempo se esgotar.

3.2.3 Circuito

A especificação do barramento CAN exige que haja uma resistência de 60 Ω entre as linhas CAN High e CAN Low. Nesse teste foi utilizado um jumper no header J1 para habilitar o resistor de terminação de 120 Ω de cada módulo.

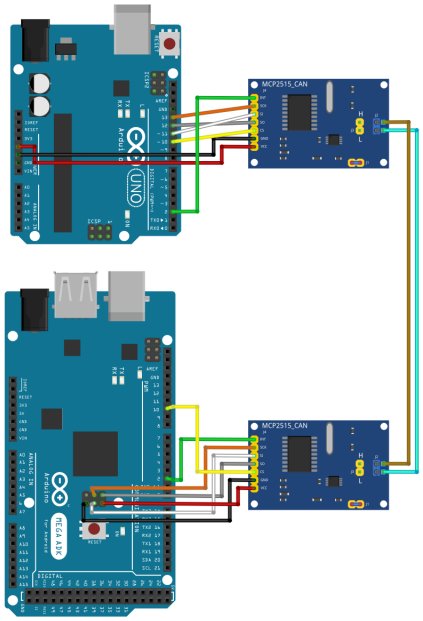

Figura 2: Ligação dos módulos e formação do barramento

3.2.4 Resultados esperados

Após 5 minutos de teste espera-se que sejam obtidas exatamente 300000 frames ($1000Hz \times 60 \times 5$).

3.3 Teste STM32 + TJA1050

O terceiro teste aproveita o conjunto MCP2515 + Arduino para criar um barramento CAN. O objetivo e recebimento de mensagens é similar ao teste anterior, mas o envio dos frames será realizado pelo periférico CAN do STM32F103C8T6 através do transceiver TJA1050

3.3.1 Circuito

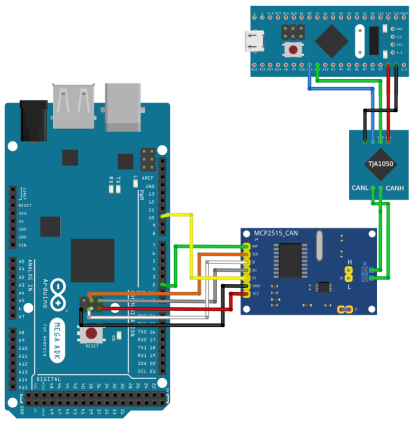

Figura 3: Ligação do módulo CAN, transceiver e microcontroladores

3.3.2 Resultados esperados

O resultado esperado para esse teste é exatamente o mesmo do teste anterior.

3.4 Teste STM32 Bidirecional

O último teste é útil para validar o envio e recebimento de mensagens CAN através dos periféricos dos microcontroladores STM e do transceiver TJA1050. O frame CAN é o mesmo do teste 3.2.

3.4.1 Comunicação serial

Não é possível utilizar simultaneamente os periféricos CAN e USB 2.0 do STM32F103C8T6. Portanto, a placa Arduino Mega 2560 será utilizada como conversor UART → USB. Essa placa específica foi escolhida simplesmente para agilizar o teste, mas sua função poderia ser igualmente cumprida por um MAX232 ou um FTDI genérico. A baud rate do sistema UART é a mesma dos testes anteriores, 115200 bps.

3.4.2 Resultados esperados

O resultado esperado para esse teste é exatamente o mesmo do item 3.2.4.

4 Resultados

4.1 Teste MCP2515 + Arduino

O teste foi montado conforme a figura abaixo:

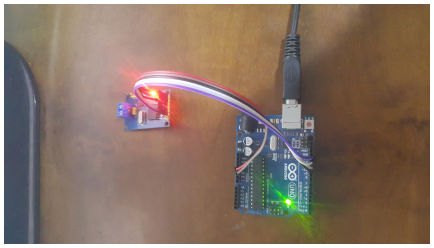

Figura 4: Circuito de teste

Utilizando o comando UNIX "cat", verificou-se a saída serial (O computador alocou o dispositivo virutal /dev/ttyACM1 para o arduino) em um terminal, obtendo o resultado:

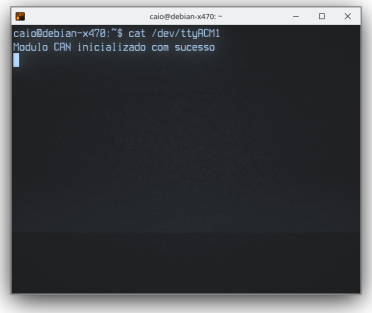

Figura 5: Resultado da inicialização

4.2 Teste MCP2515 Bidirecional

O teste foi montado conforme a figura abaixo:

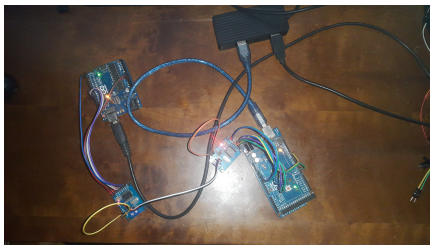

Figura 6: Circuito de teste

Utilizando o programa "validaSerial.js", verificou-se que a quantidade de mensagens enviadas num período de 5 minutos foi de 300002. As mensagens extra aconteceram devido à pequenas diferenças entre o timer do microcontrolador e o RTC do computador, mas ao verificar os timestamps foi possível constatar que não houve perda de dados.

4.3 Teste STM32 + MCP2515

O teste foi montado conforme a figura abaixo:

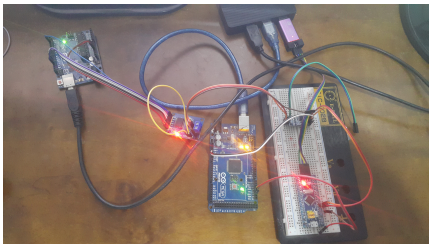

Figura 7: Circuito de teste

O programa anterior foi utilizado para validar o circuito com o stm32 e o transceiver TJA1050. Foram recebidas 300004 mensagens em 5 minutos, diferença explicada pelo mesmo motivo do último teste.

4.4 Teste STM32 Bidirecional

O teste foi montado conforme a figura abaixo:

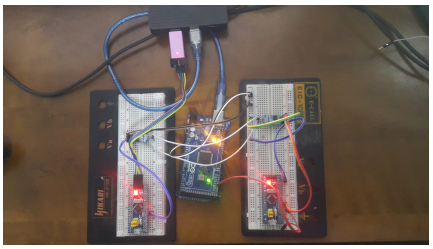

Figura 8: Circuito de teste

Após a validação, foi constatado o envio de 300004 mensagens em 5 minutos. Utilizou-se também um "console.log()" para visualizar todos os dados do frame:

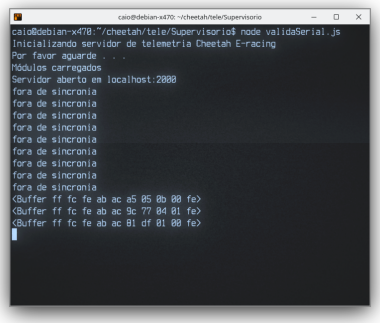

Figura 9: Buffer serial recebido do stm32

5 Conclusões

Todos os testes realizados obtiveram resultados positivos. Esse é um passo importante para a implementação do barramento CAN no sistema de telemetria do Cheetah E-Racing com confiabilidade e rapidez, e abre caminho para o desenvolvimento de placas de aquisição de dados com todos os benefícios que o protocolo proporciona.