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

Teste da comunicação CAN - BMS

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

Resumo

A comunicação CAN com o BMS (Battery Management System) é fundamental para fornecer as informações do acumulador para o subsistema de telemetria, e portanto garantir o seu funcionamento é uma das prioridades da subequipe.

1 Objetivo

O objetivo do teste é verificar a comunicação unidirecional do BMS − > Adapatador CAN MCP2515 que será utilizado na placa de telemetria traseira do carro.

2 Equipamentos

Para os testes foram utilizados:

- 1 Orion BMS;
- 1 Conector MainIO TE 1376360-1 de 26 pinos;
- 1 Cabo DB9 Fêmea;
- 1 CANAdapter + Cabo USB tipo A-B;
- 1 Placa de desenvolvimento Arduino com microcontrolador ATmega2560;
- 1 Módulo CAN MCP2515 + TJA1050;
- Componentes passivos e jumpers de protoboard.

3 Procedimentos

3.1 Configuração do BMS

As mensagens CAN devem ser configuradas no BMS utilizando o software proprietário da empresa. Foi utilizado o método de broadcast na interface CAN1, a uma taxa de 500kbps. A frequência de envio configurada foi de 10Hz ou 100ms entre mensagens. Como referência, utilizou-se a tabela de prioridade CAN definida pela equipe na nossa documentação, disponível em: `https://wikicheetah.readthedocs.io/en/latest/intro.html#comunicacao-can`

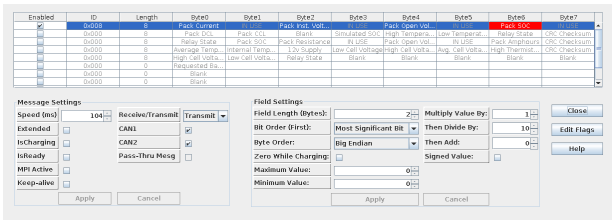

Figura 1: Configuração no Orion BMS Utility

3.2 Programação do BMS

Para programar o BMS, utilizou-se o CANAdapter para baixar as configurações atuais, alterá-las e refazer o envio. O adaptador foi conectado na interface CAN1 com o auxílio de um conector DB9 Fêmea.

3.3 Conexão BMS - MCP2515

Finalmente, para realizar o teste de fato conectaram-se as linhas CAN High e CAN Low do OrionBMS no módulo MCP2515, com o cuidado de configurar jumper do módulo para habilitar o resistor de terminação de 120 Ω.

3.4 Código de teste

Utilizou-se a biblioteca Cheetah (disponível no GitHub da equipe) para realizar a comunicação CAN, e o sistema supervisório (também disponível no nosso GitHub) para visualizar o recebimento das informações. Devido à COVID-19, o tempo de oficina é restrito e portanto realizou-se a configuração de apenas um parâmetro, o SoC, no dashboard principal do supervisório. O código utilizado para o teste encontra-se nos anexos desse documento.

4 Resultados

O resultado da comunicação pôde ser observado na interface de usuário sistema supervisório e no log de console do node.js. Devido ao tempo curto disponível, a obtenção de gravações dessa interface funcionando não foi priorizada, porém o resultado obtido foi semelhante ao da figura abaixo:

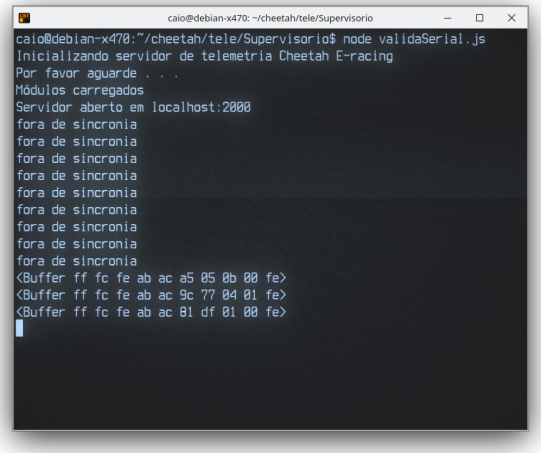

Figura 2: Representação do resultado do teste

5 Código de teste

```
/* Código fonte para teste de comunicação CAN - BMS
 * Não esquecer de configurar os pinos no config.h
 * Plataforma: ATmega2560
 * Autor: Caio Tácito
 * Data: 03/05/2021
 * CHEETAHPORRA!
*/
#include "Cheetah.h"

//Instanciando objetos
CheetahCAN can;
CheetahSerial serial;

void setup()
{
  //Incializa serial
  Serial.begin(115200);
  //Inicializa CAN
  can.beginCAN();
}

void loop()
{
  //Verifica se há mensagem disponível
  if(can.readMessage())
  {
    //Guarda a mensagem em um vetor
    byte* dados = can.getMsg();
    //Converte os valores para 8 bits shiftando e fazendo OR
    //E depois os adiciona ao buffer serial
    serial.addAnalogSensor(dados[0] | dados[1]<<8);
    serial.addAnalogSensor(dados[2] | dados[3]<<8);
    serial.addAnalogSensor(dados[4] | dados[5]<<8);
    serial.addAnalogSensor(dados[6] | dados[7]<<8);
    //Envia os dados para a serial
    serial.sendPayload();
  }
}
```