INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

ENGENHARIA MECATRÔNICA

INFORMÁTICA INDUSTRIAL II

Manual de utilização de Protocolo CAN Bus para comunicação entre Arduinos com módulo CAN MCP2515

Gustavo Belmonte Cioccari

Pablo Medeiros Penna

Florianópolis, Novembro de 2019

1. **O projeto**

Nesse projeto foi realizada a comunicação entre 4 arduinos utilizando o protocolo CAN Bus. O objetivo era através de um potenciômetro, um botão e 2 LEDs representar o freio e o pisca-alerta de um carro. O potenciômetro simula a intensidade do freio, sendo evidenciada em um LED e o botão simula o acionamento do pisca alerta, representado também por um LED. Além disso foi desenvolvido um sistema SCADA que se comunica com um Arduino “central”, responsável por interpretar os sinais enviados pelos outros Arduinos.

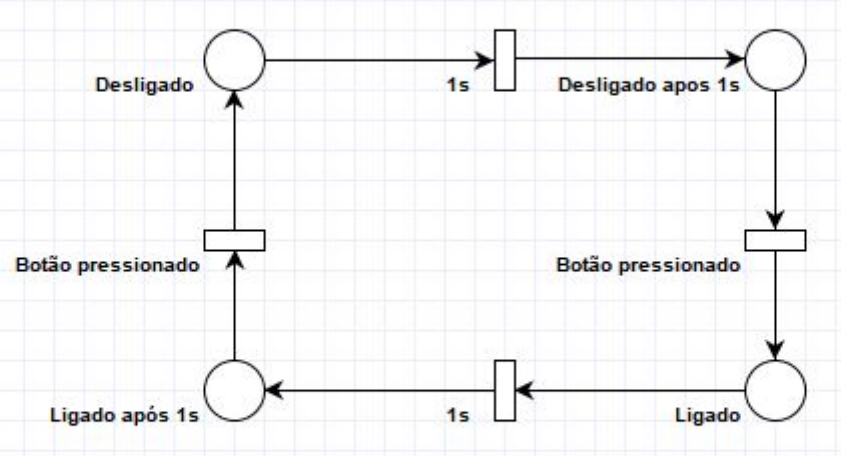
2. **Módulo CAN**

Para utilização do protocolo CAN Bus com Arduino foram utilizados módulos CAN MCP2515 que implementam a versão 2.0B e tem velocidade de transmissão de 1Mbps. Para realizar a comunicação é necessário um módulo para cada Arduino que se deseja comunicar.

3. **Camada de aplicação**

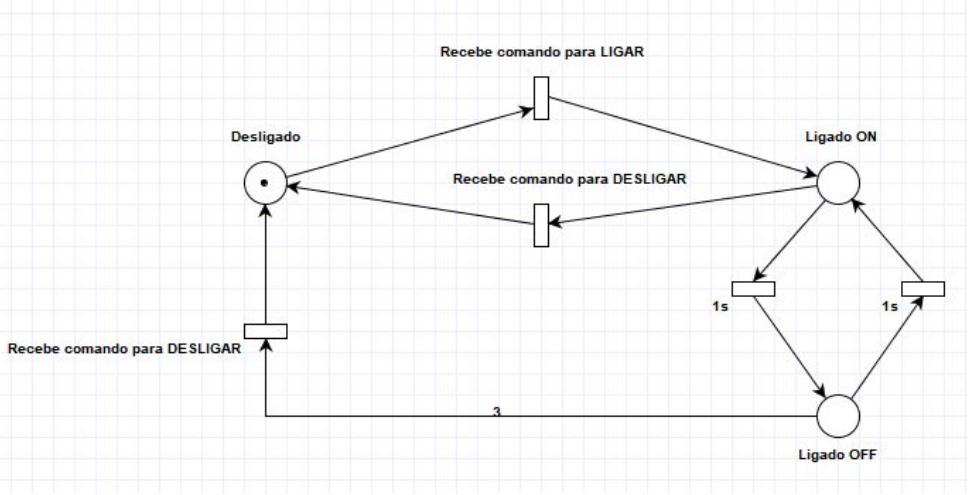
Para implementar a comunicação foi utilizada a biblioteca CAN_BUS_Shield para Arduino. Essa biblioteca é um projeto aberto no GitHub e está disponível no aqui. Essa biblioteca contém funções que facilitam a programação para o estabelecimento de conexão, envio, recebimento e leitura dos sinais. Os códigos desenvolvidos para a aplicação encontram-se no repositório do GitHub disponível aqui e possuem comentários para um fácil entendimento. Para criar o sistema de pisca, foi construída uma máquina de estados já que foi utilizado um *push button*, dessa forma era necessário saber o estado anterior para determinar se o pisca deve ligar ou desligar. O Arduino que envia o sinal do botão só envia esse sinal uma vez, sendo 1 para ligar e 0 para desligar o pisca, portanto o Arduino que recebe o sinal sabe que quando recebe 1 deve piscar o LED e quando recebe 0 deve manter o LED desligado. As máquinas de estados criadas encontra-se na figura 1.

Figura 1: Máquina de estados botão do pisca



Fonte: Os autores (2019)

Figura 2: Máquina de estados LED do pisca

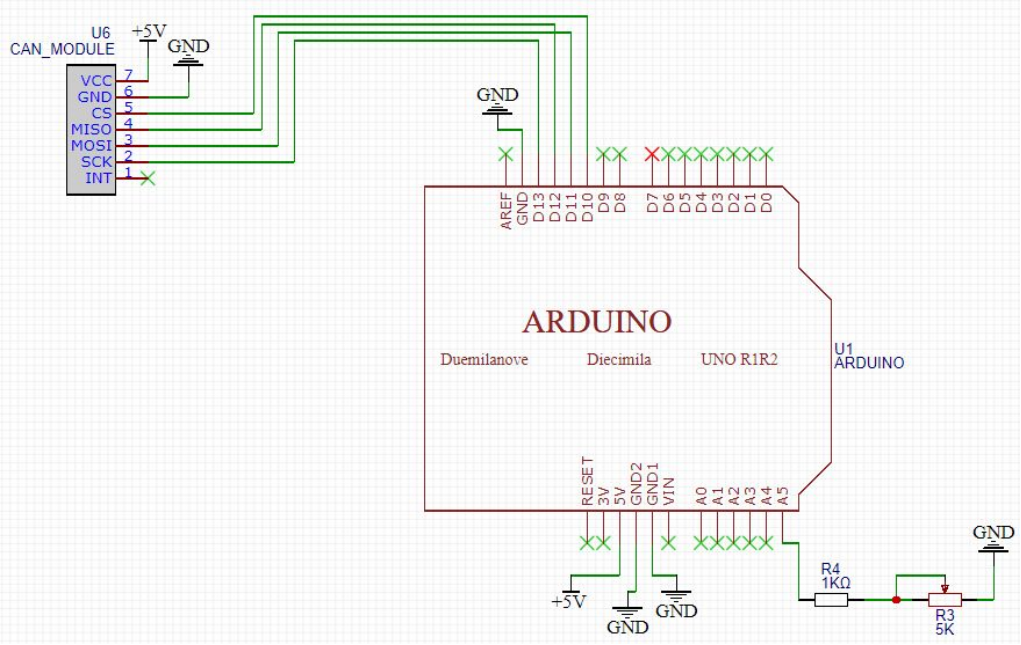


Fonte: Os autores (2019)

4. Circuito eletrônico

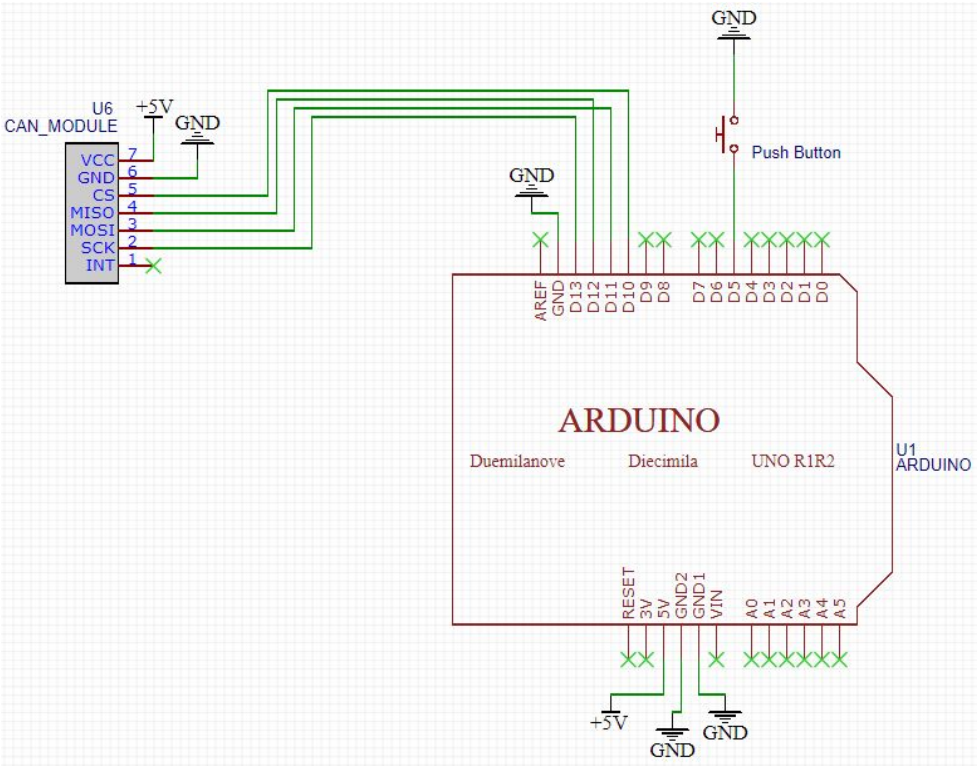
Foram elaborados três circuitos eletrônicos, todos eles contendo um Arduino e um módulo CAN MCP2515. Os esquemas eletrônicos de cada circuito tanto os emissores quanto os receptores estão disponíveis nas imagens abaixo, além do esquema de ligação entre os módulos CAN. Os resistores de pull-up que eram necessários foram utilizados os do próprio Arduino, habilitados por código.

Figura 3: Circuito eletrônico emissor potenciômetro (freio)



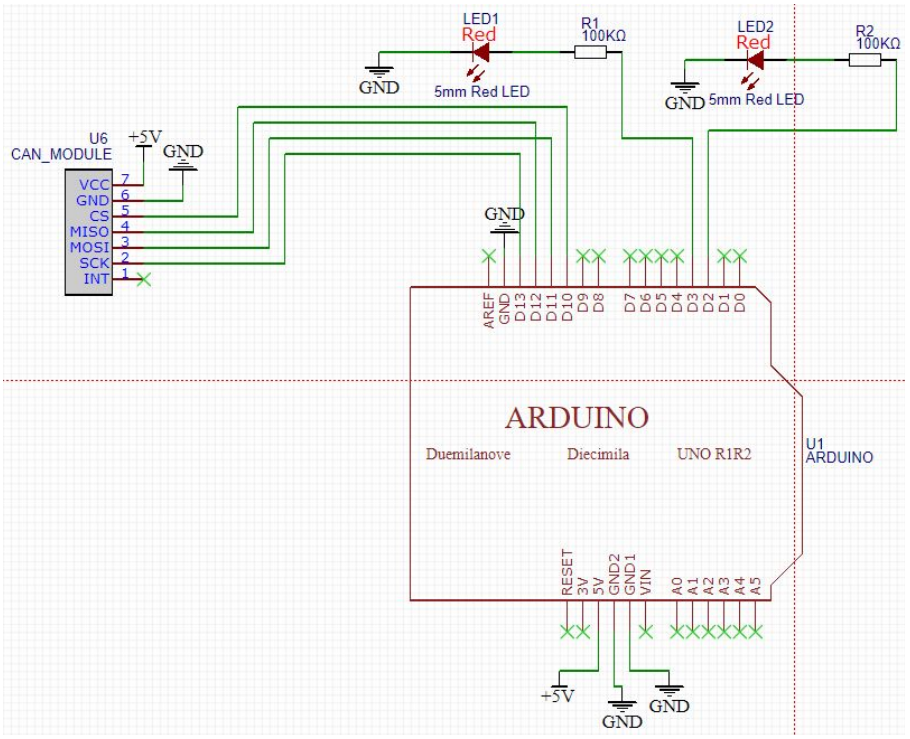
Fonte: Os autores (2019)

Figura 4: Circuito eletrônico emissor botão (pisca)



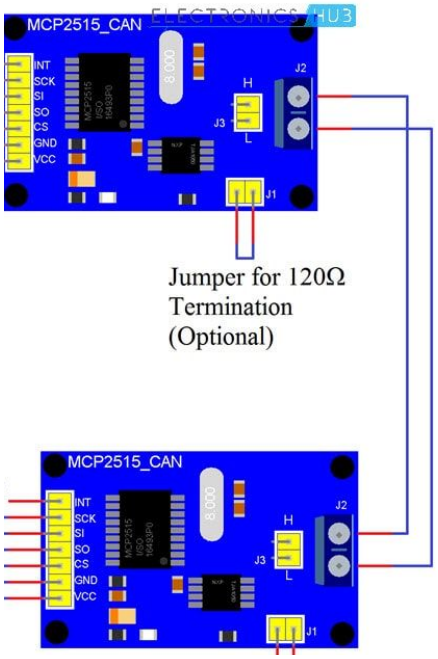
Fonte: Os autores (2019)

Figura 5: Circuito eletrônico receptor LEDs



Fonte: Os autores (2019)

Figura 6: Ligação entre módulos CAN

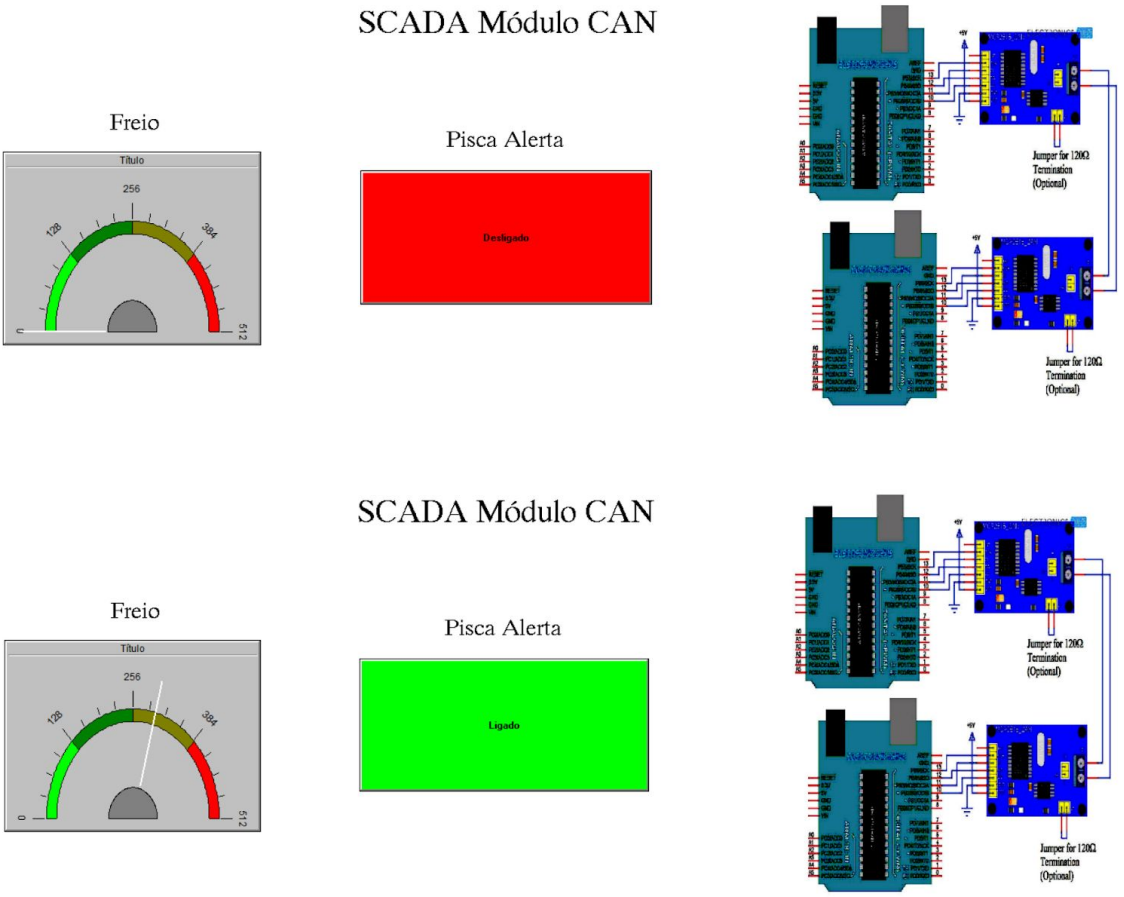


Fonte: Electronics Hub (2018)

5. SCADA

Para monitorar o sistema foi criado um SCADA no software Elipse SCADA, utilizando a biblioteca chamada Modbusino para estabelecer a comunicação entre o Arduino central, responsável por receber todas as mensagens e repassar ao SCADA via Modbus. O sistema desenvolvido encontra-se na figura 7.

Figura 7: Sistema SCADA

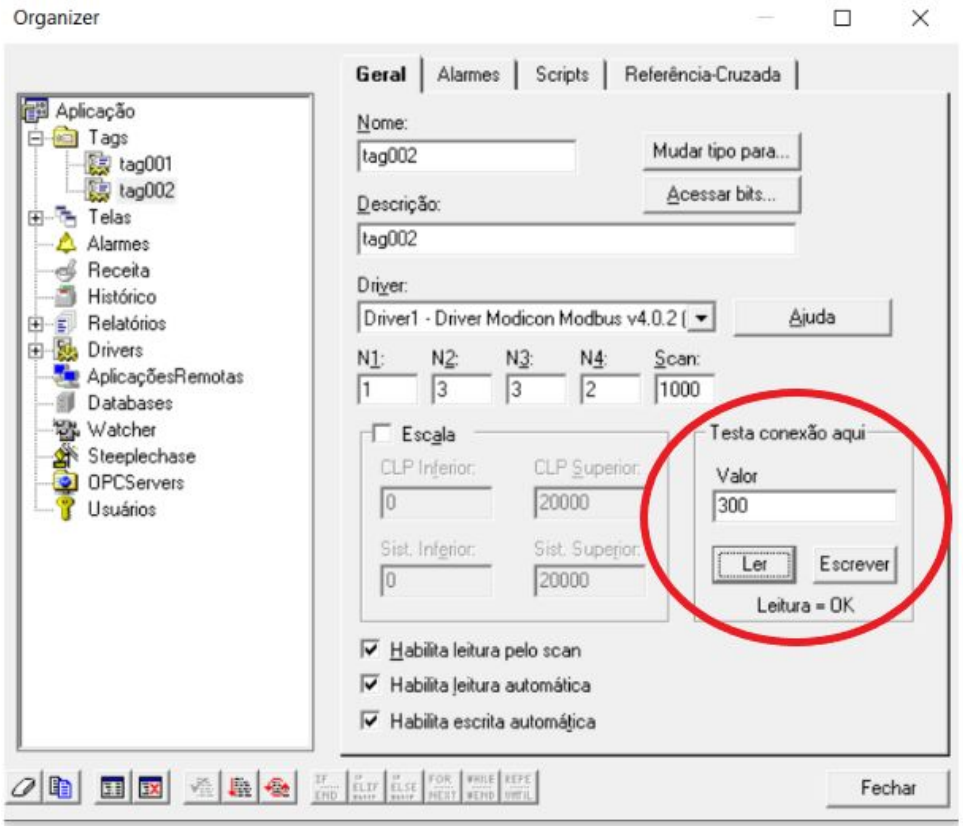


Fonte: Os autores (2019)

Em uma possível aplicação real, o gráfico representaria a intensidade com que o freio foi acionado e o quadro do pisca informaria o estado do pisca, ligado ou desligado.

A comunicação com sucesso entre o Arduino central e o SCADA pode ser vista na figura 8.

Figura 8: Comunicação Arduino e SCADA



Fonte: Os autores (2019)

6. Aprendizados

Com esse projeto pudemos entender na prática como funciona a comunicação CAN Bus e constatar que pode ser utilizado em diversos tipos de aplicações nas mais diversas áreas. Além disso serviu para esclarecer alguns pontos como a prioridade, que entendemos que ocorre na camada física e não na aplicação. Esclareceu a forma que ocorre a comunicação e como cada Arduino “filtra” as mensagens enviadas, se devem ser assimiladas por ele ou não.