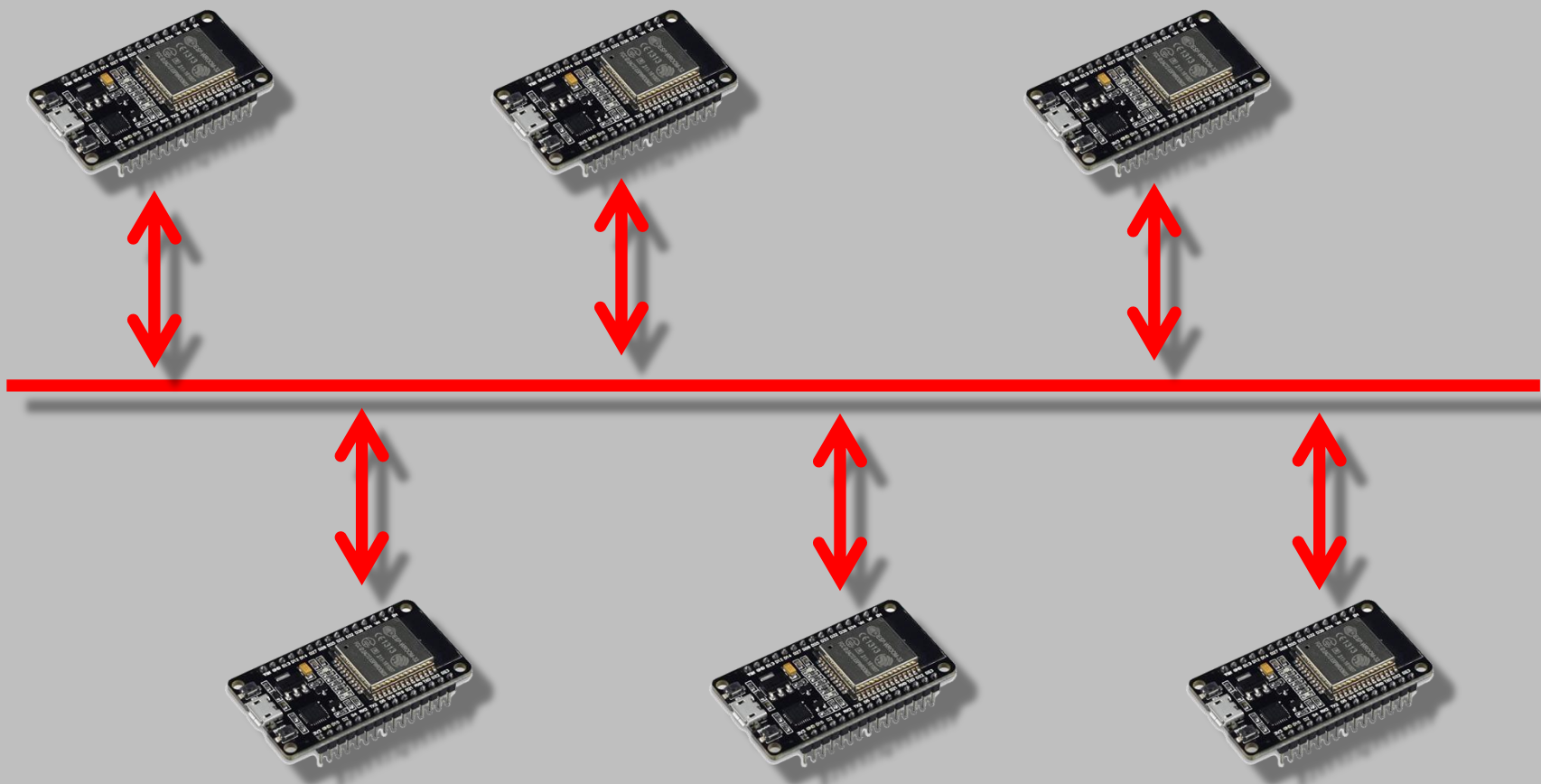


Introdução ao protocolo CAN com ESP32

CAN



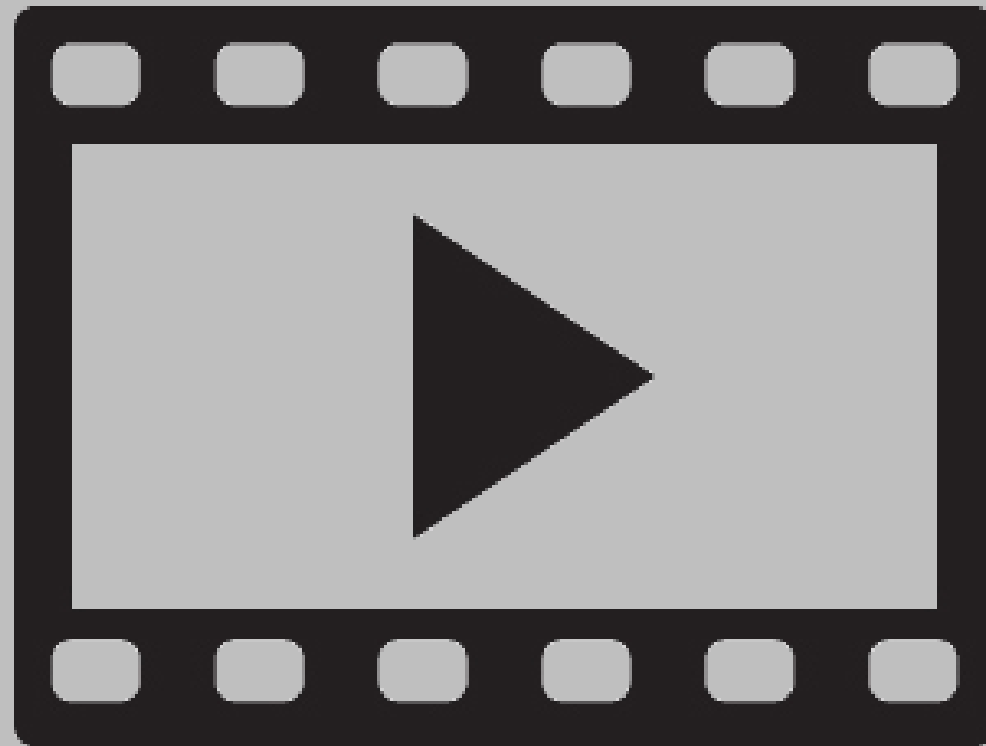
Por Fernando Koyanagi

Intenção dessa aula

- 1. Introduzir os conceitos básicos do protocolo CAN**
- 2. Realizar uma montagem simples com ESP32 usando o protocolo CAN**



Demonstração



Recursos usados

- Dois módulos ESP WROOM 32 NodeMcu
- Dois módulos transceivers CAN da WaveShare
- Jumper's para conexões
- Analisador lógico para captura
- Três cabos USB para os ESP's e analisador
- 10 metros de par trançado para servir de barramento



CAN (Controller Area Network)

- Desenvolvido pela Robert Bosch GmbH nos anos de 1980 para atender a indústria automobilística.
- Tornou-se muito difundida com o passar dos anos devido à sua robustez e flexibilidade de implementação, sendo introduzida em equipamentos militares, máquinas agrícolas, na automação industrial e predial, robótica e equipamentos médicos.



BOSCH

Invented for life



Instagram

fernandok_oficial

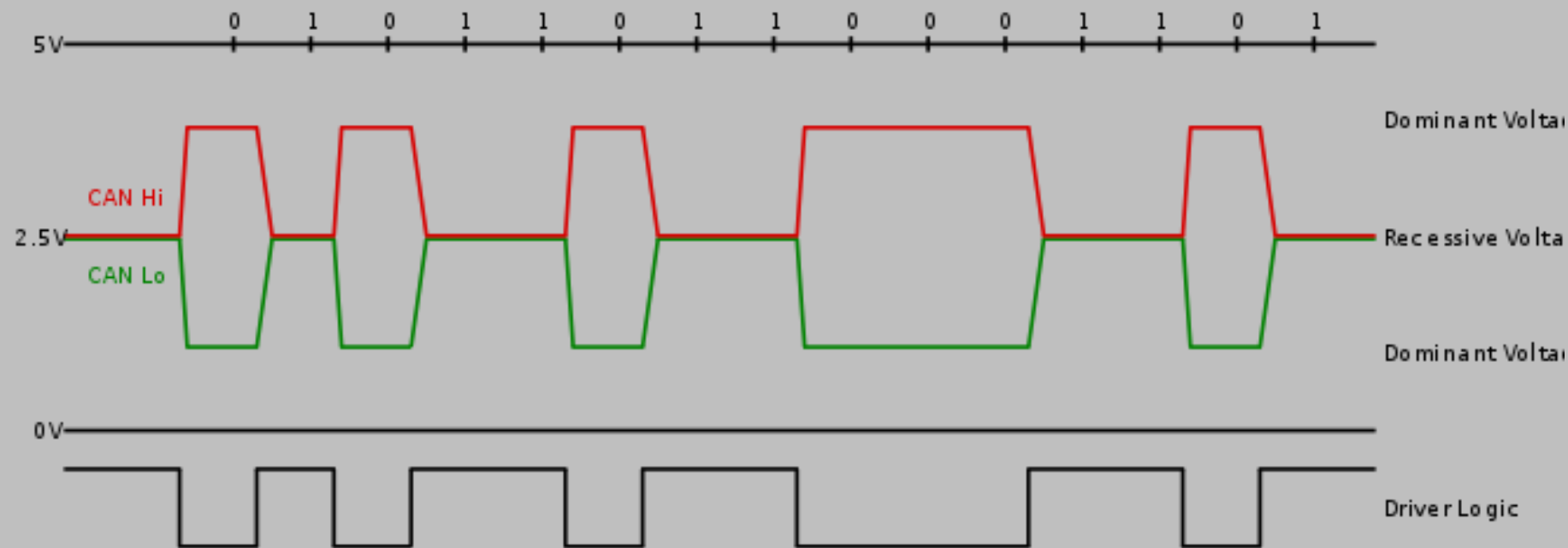


Telegram

fernandok_oficial



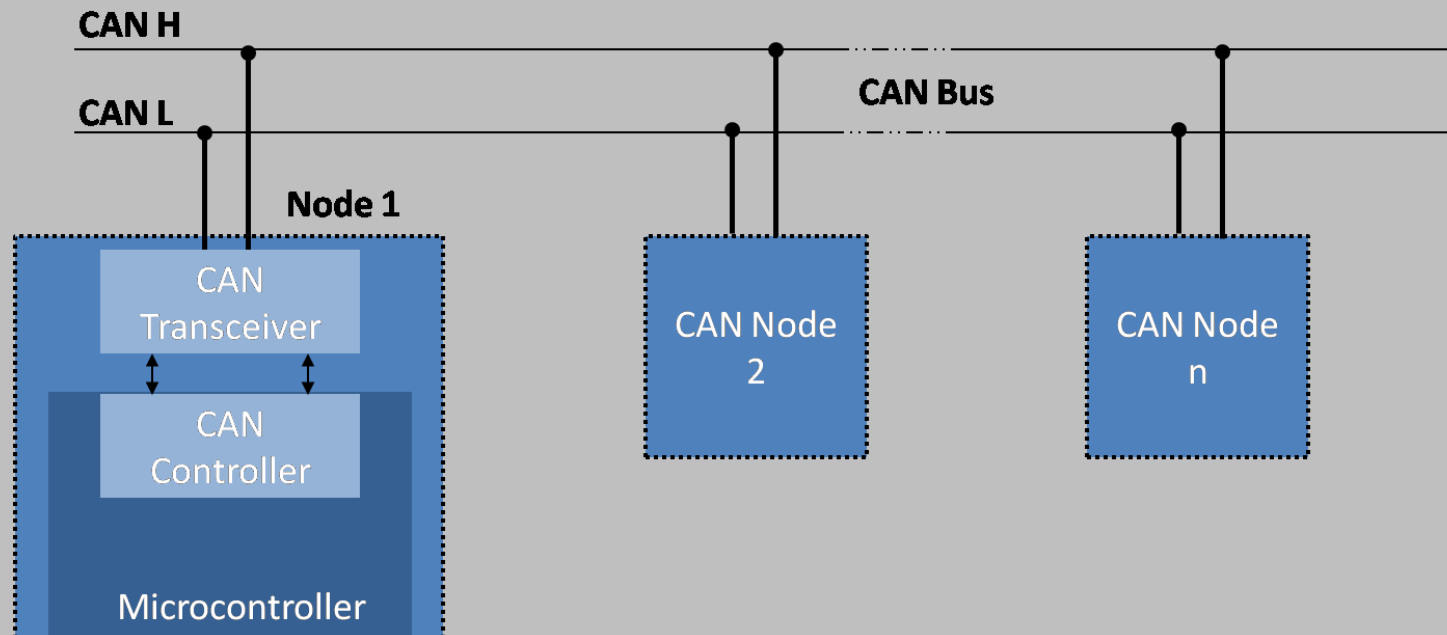
CAN - Algumas características



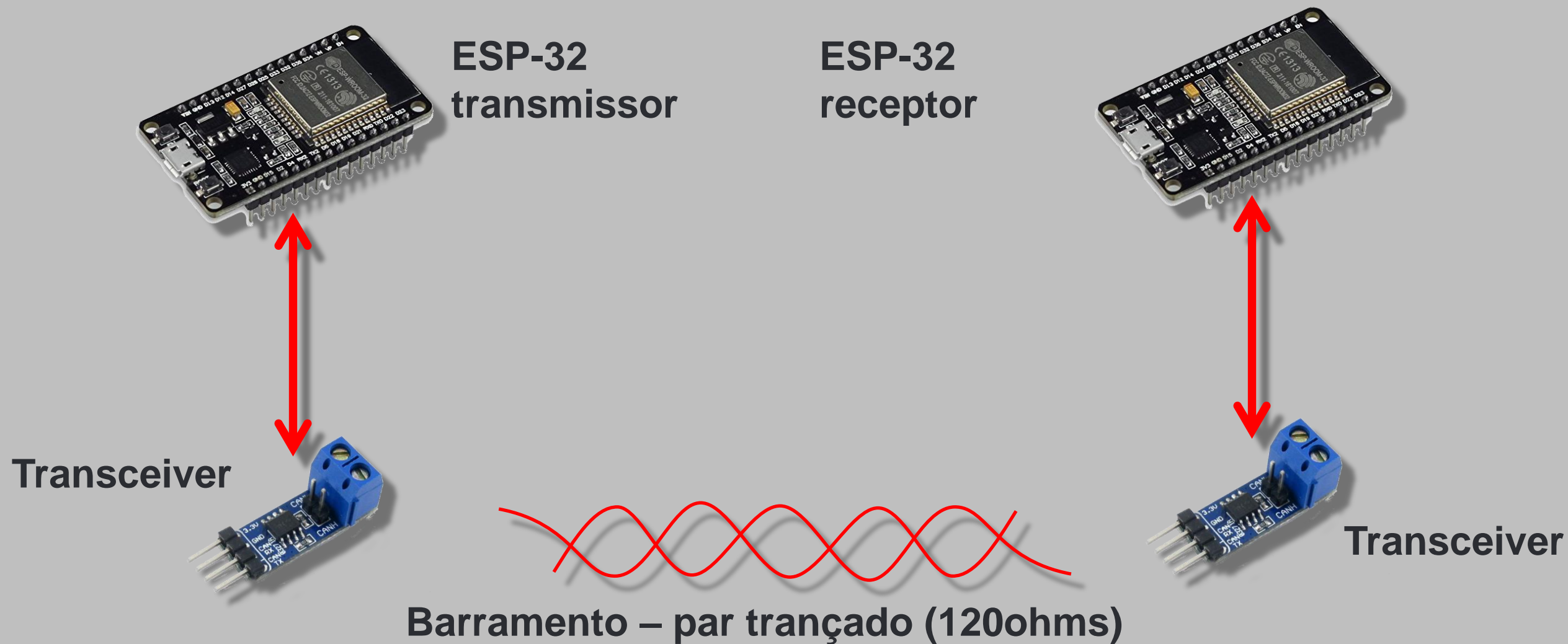
- Comunicação serial de dois fios
- Máximo de 8 bytes de informação útil por quadro, sendo possível fragmentação.
- Endereçamento direcionado a mensagem e não ao nó.
- Atribuição de prioridade às mensagens e retransmissão de mensagens “em espera”
- Capacidade eficaz de detectar e sinalizar erros.
- Capacidade multi-mestre (todos os nós podem pedir acesso ao barramento)
- Capacidade multicast (uma mensagem para vários receptores ao mesmo tempo)

CAN - Algumas características

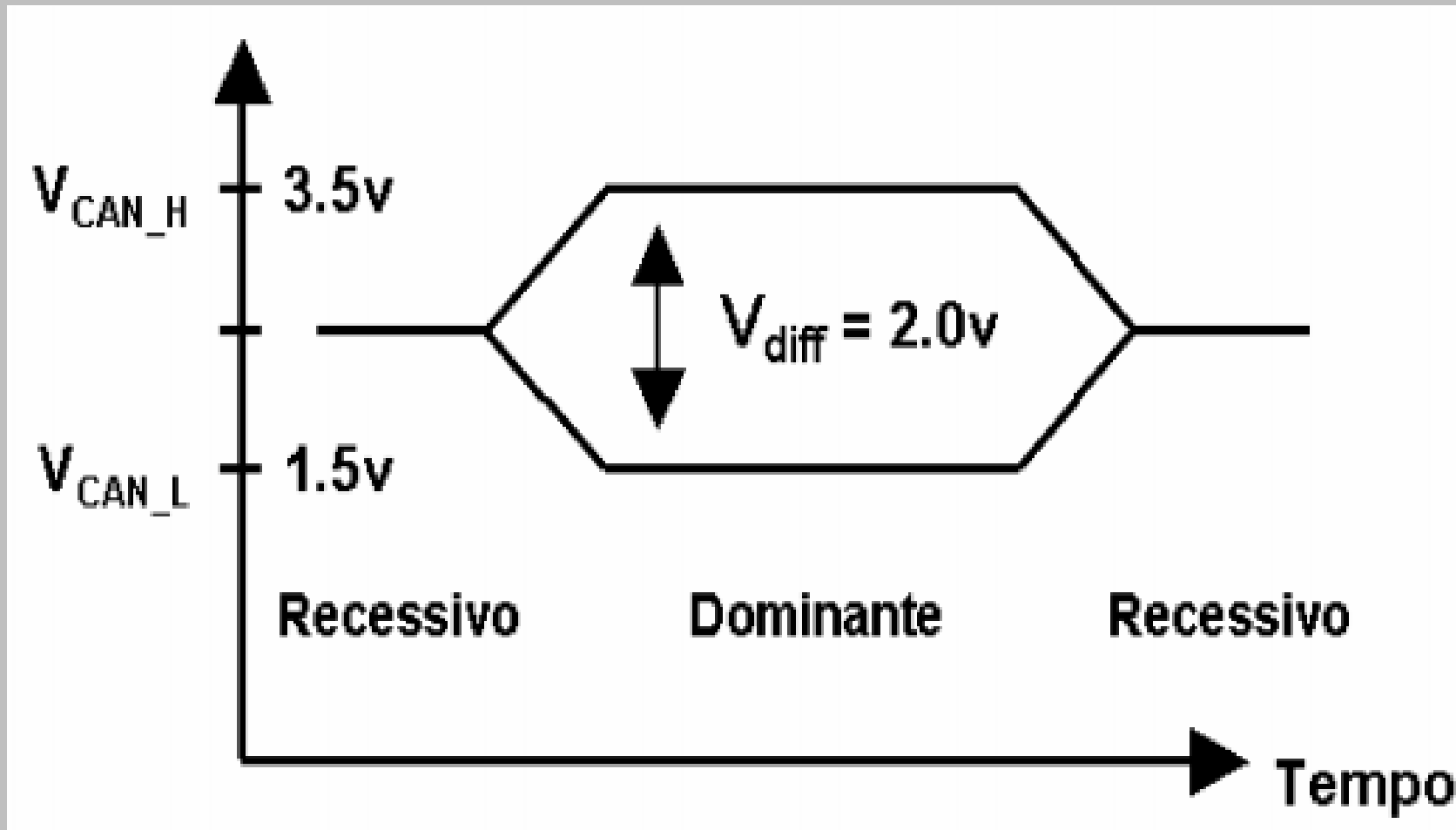
- Taxas de transferências de até 1Mbit/s em barramento de 40 metros. (Redução da taxa com aumento do comprimento do barramento).
- Flexibilidade de configuração e introdução de novos nós (comportando até 120 nós por barramento).
- Hardware padrão, baixo custo e boa disponibilidade.
- Protocolo regulado: ISO 11898



Circuito utilizado



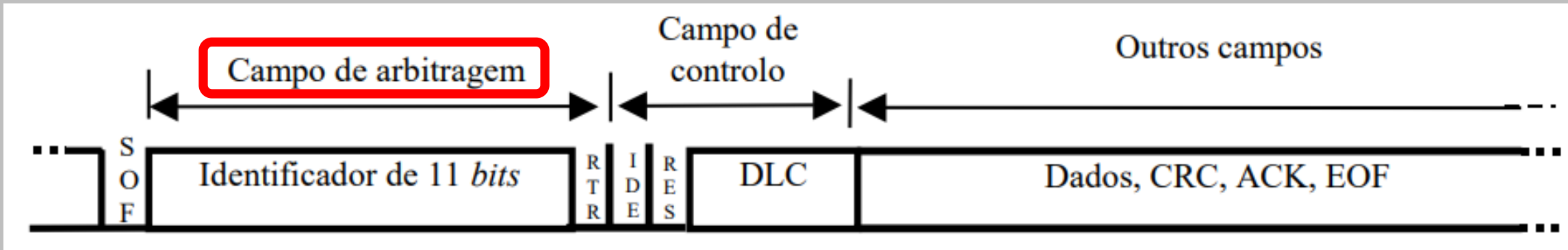
Tensões na linha de transmissão (detecção diferencial)



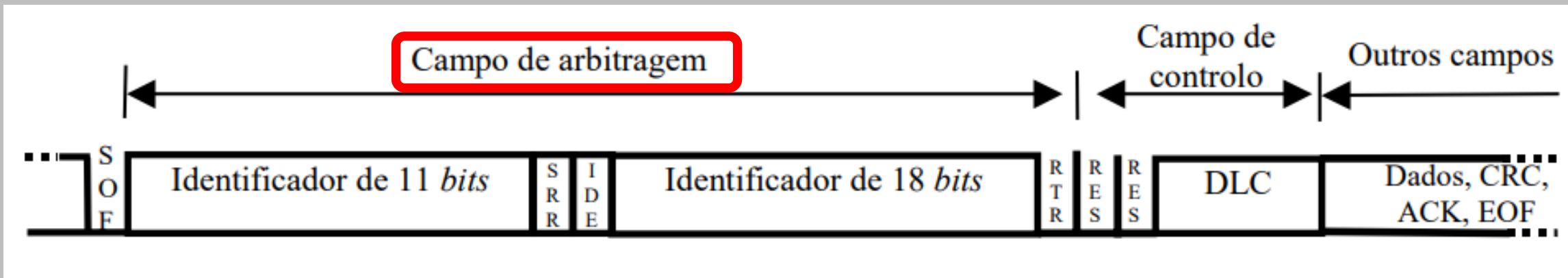
No CAN o bit dominante é o Zero.

Detecção diferencial na linha reduz a sensibilidade a ruídos (EFI)

Padrões de CAN e formato das frames



Formato padrão com identificador de 11 bits

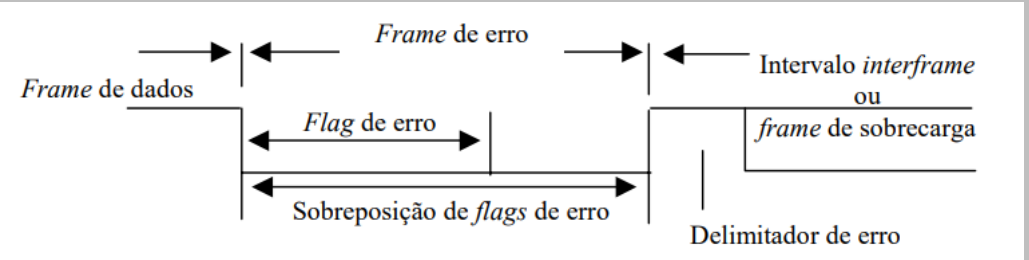
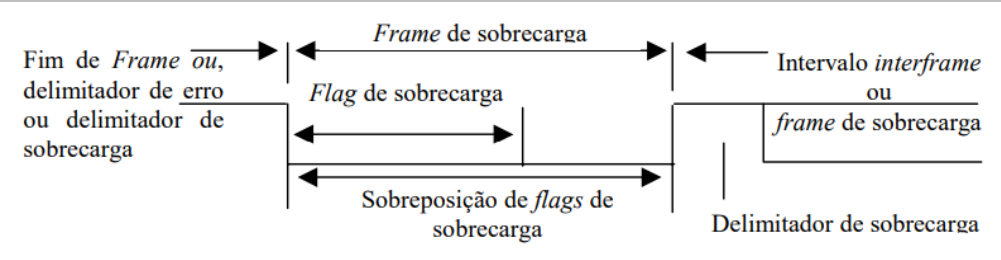
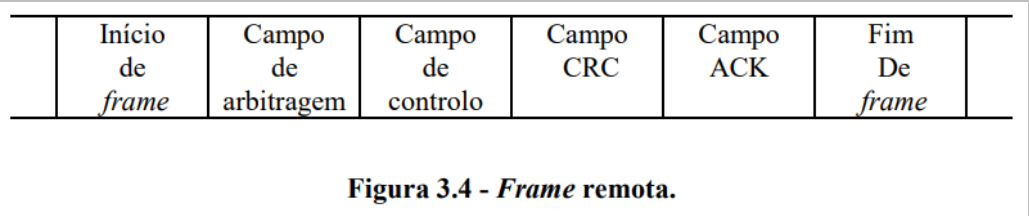
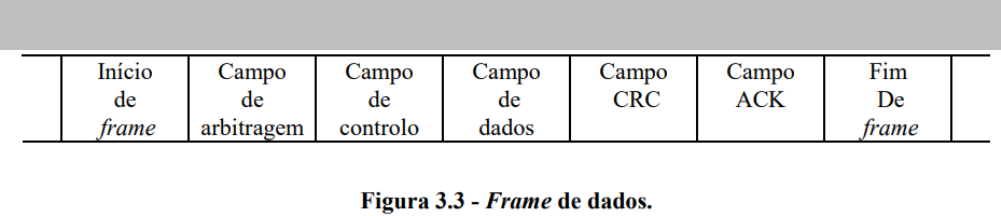


Formato estendido com identificador de 29 bits

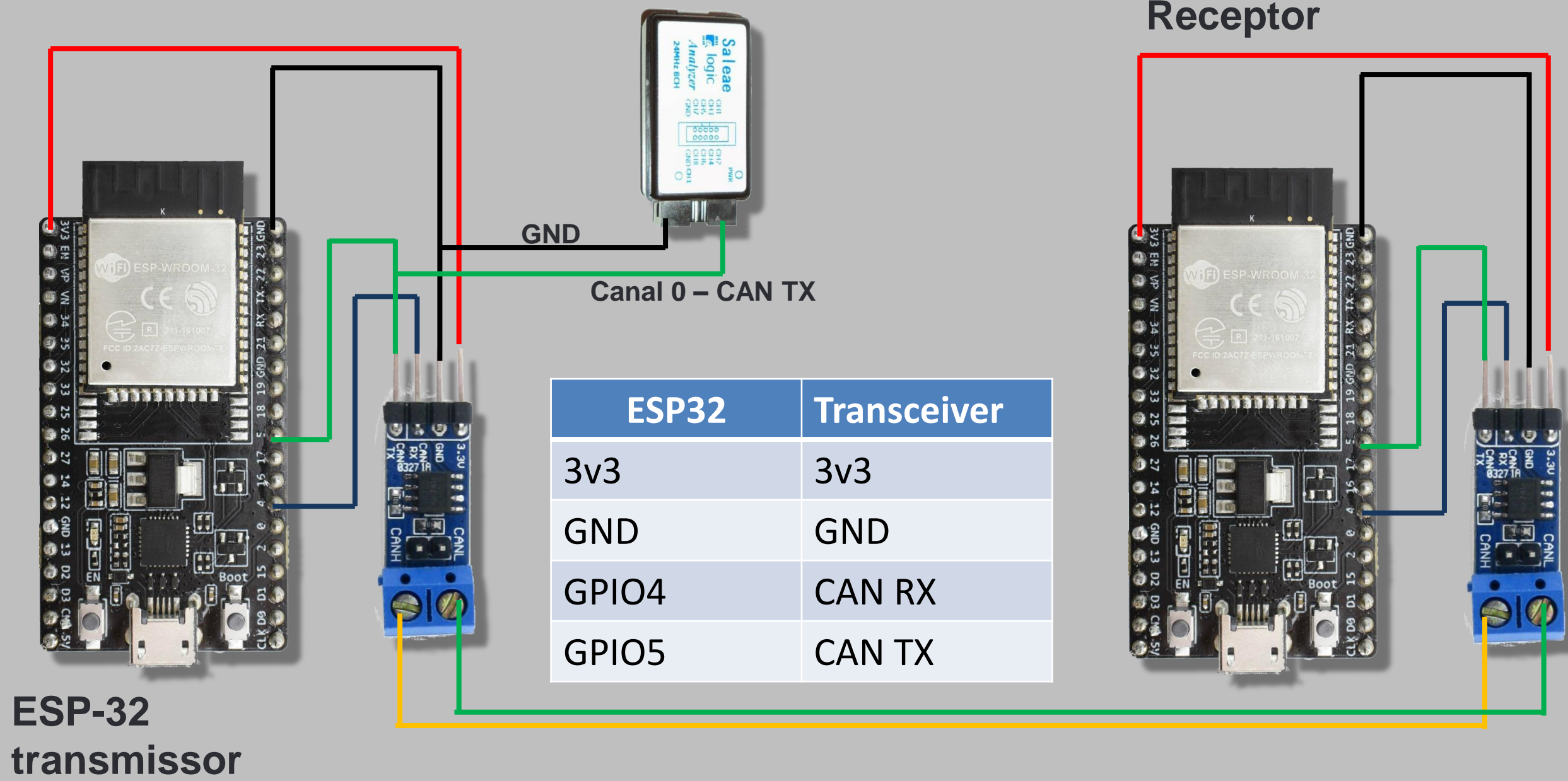
Quatro tipos de frames (quadros)

A transmissão e recepção de dados no CAN é baseada em quatro tipos de frames. Os tipos de frames serão identificados pelas variações nos bits de controle ou até por mudanças nas regras de escrita do frame para cada caso.

- **Frame de Dados:** Contem os dados do emissor para o(s) receptor(es)
- **Frame Remota:** É uma solicitação de dados partindo de um dos nós
- **Frame de Erro:** É um frame enviado por qualquer um dos nós ao identificar um erro no barramento e pode ser detectado por todos os nós
- **Frame de sobrecarga:** Serve para retardar o trafego no barramento devido à sobrecarga de dados ou atraso em um ou mais nós.

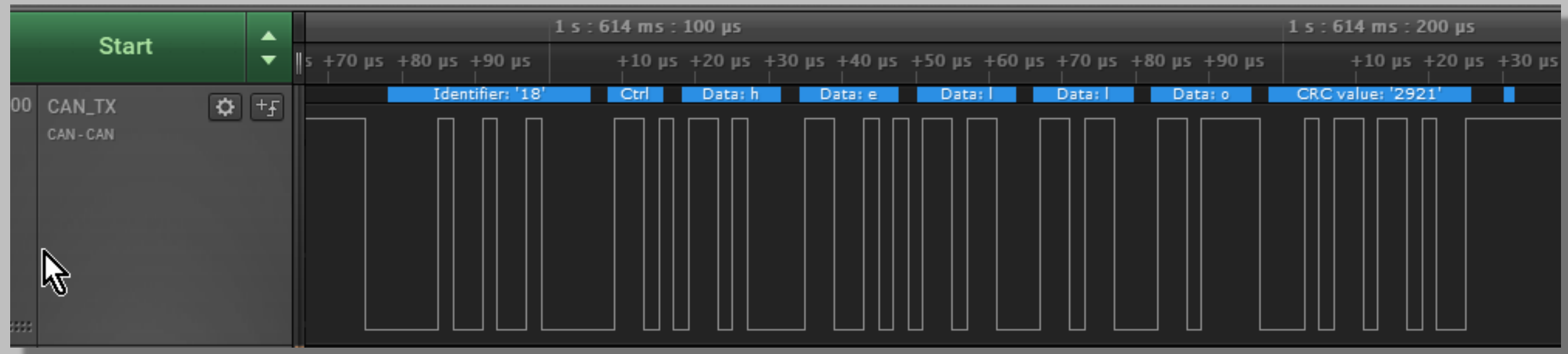


Circuito – detalhe das conexões

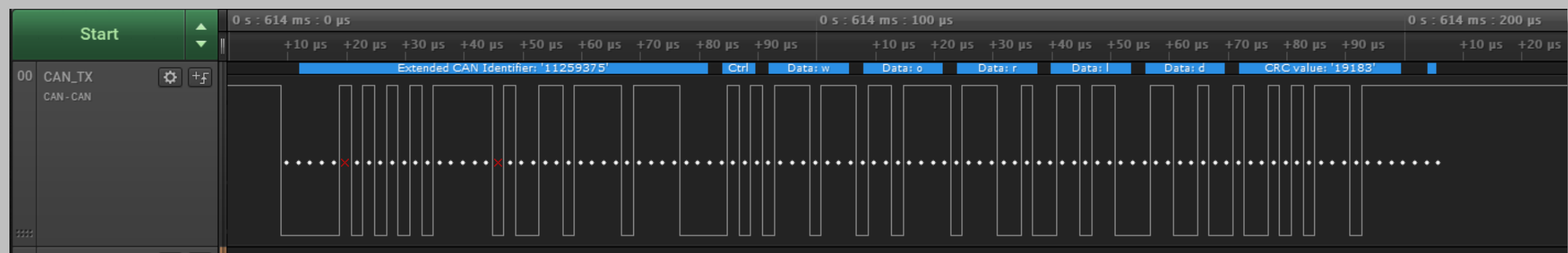


Circuito – Captura dos dados

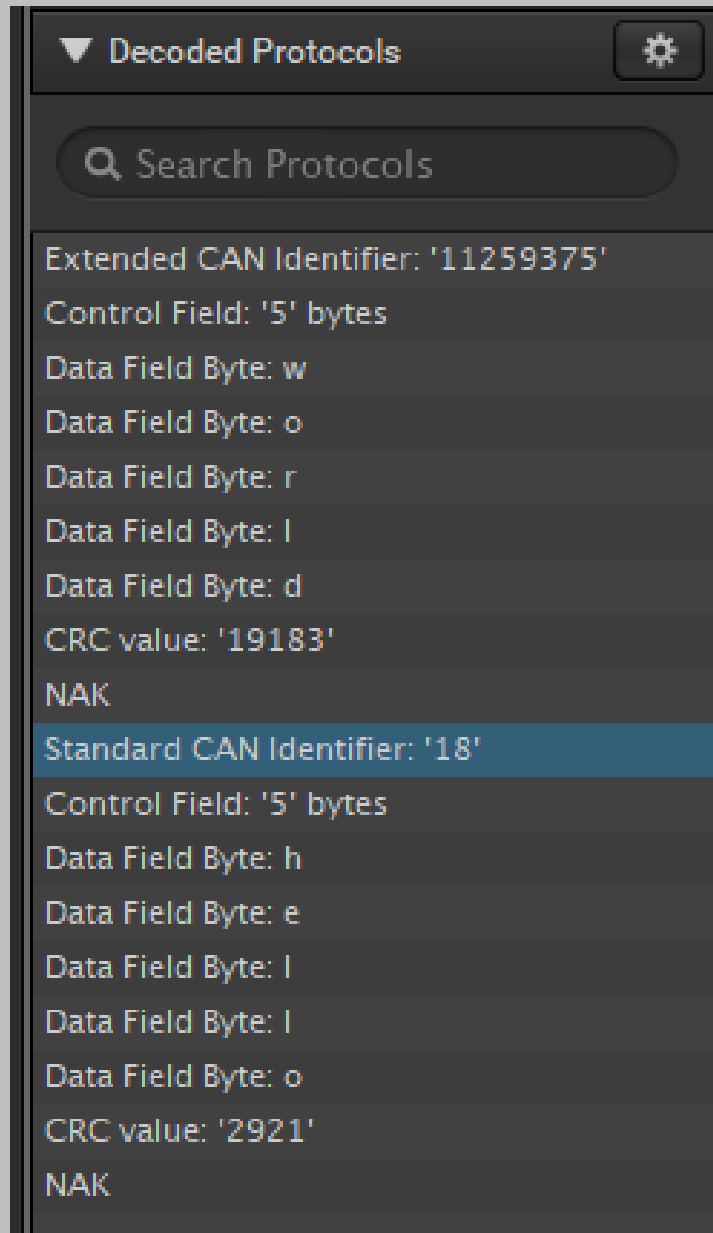
Formas de onda obtidas para CAN padrão com ID de 11 bits



Formas de onda obtidas para CAN estendido com ID de 29 bits




Circuito – Captura dos dados



Dados obtidos pelo analisador lógico

Biblioteca Arduino - CAN

Biblioteca para controlador CAN

 Gerenciador de Biblioteca

Tipo Todos

Tópico Todos

CAN


CAN by Sandeep Mistry Versão 0.3.0 **INSTALLED**
An Arduino library for sending and receiving data using CAN bus. Supports Microchip MCP2515 based boards/shields and the Espressif ESP32's built-in SJA1000 compatible CAN controller.
[More info](#)

CAN-BUS Shield by Seeed Studio
Arduino library to control CAN-BUS Shield. Arduino library to control CAN-BUS Shield.
[More info](#)

CapacitiveSensor by Paul Bagder, Paul Stoffregen
Create capacitive sensors that can detect touch or proximity. The capacitiveSensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.
[More info](#)

Fechar

Biblioteca para controlador CAN

 Features Business Explore Marketplace Pricing

Search / Sign in or Sign up

sandeepmistry / arduino-CAN

Watch 7 Star 17 Fork 3

Code Issues 1 Pull requests 0 Projects 0 Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

An Arduino library for sending and receiving data using CAN bus.

32 commits 1 branch 4 releases 2 contributors MIT

Branch: master New pull request Find file Clone or download

sandeepmistry Version 0.3.0 Latest commit a75d52a 22 days ago

examples

Add license and comments

6 months ago

src

Move MCP2515_DEFAULT_CLOCK_FREQUENCY outside

22 days ago

<https://github.com/sandeepmistry/arduino-CAN>



Em www.fernandok.com

Seu e-mail



PRINCIPAL SOBRE FERNANDO K ARDUINO ESP8266 ESP32 LORAWAN MOTOR DISPLAY MATERIAIS DOWNLOAD

Receba o meu conteúdo
GRATUITAMENTE

Insira aqui seu melhor email...

QUERO RECEBER GRÁTIS



Motor de Passo Nema 23 com Driver TB6600 e Arduino Due

by **Fernando K Tecnologia** - 2:44 PM

Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...

Leia mais



ESP32 Longa Distância - LoRaWan

by **Fernando K Tecnologia** - 9:46 AM

Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...

Leia mais



Motor de HD com Arduino

by **Fernando K Tecnologia** - 2:00 PM

QUAL ASSUNTO VOCÊ TEM

- ☐ Arduino
- ☐ ESP8266
- ☐ ESP32
- ☐ Motor
- ☐ Display
- ☐ Sensor

You may select multiple answers.

[Exibir resultados](#)

Votos até o momento: 32

Dias restantes para votar: 49

FACEBOOK



Código-fonte do Transmissor

Transmissor

Código-fonte: Includes e Setup()

CANSender.ino x

```
1  #include <CAN.h> //Inclui a biblioteca CAN
2
3  void setup() {
4      Serial.begin(9600); //inicia a serial para debug
5      while (!Serial);
6
7      Serial.println("Transmissor CAN");
8
9      // Inicia o barramento CAN a 500 kbps
10     if (!CAN.begin(500E3)) {
11         Serial.println("Falha ao iniciar o controlador CAN"); //caso não seja possível iniciar o controlador
12         while (1);
13     }
14 }
```

Transmissor

Código-fonte: Loop(), enviando um pacote CAN 2.0 padrão

```
16 void loop() {
17     // Usando o CAN 2.0 padrão
18     //Envia um pacote: o id tem 11 bits e identifica a mensagem (prioridade, evento)
19     //o bloco de dados deve possuir até 8 bytes
20     Serial.println("Enviando pacote...");
21
22     CAN.beginPacket(0x12); //id 18 em hexadecimal
23     CAN.write('h'); //1º byte
24     CAN.write('e'); //2º byte
25     CAN.write('l'); //3º byte
26     CAN.write('l'); //4º byte
27     CAN.write('o'); //5º byte
28     CAN.endPacket(); //encerra o pacote para envio
29
30     Serial.println("Enviado.");
31
32     delay(1000);
33 }
```

Transmissor

Código-fonte: Loop(), enviando um pacote CAN 2.0 estendido

```
33
34 //Usando CAN 2.0 Estendido
35 //Envia um pacote: o id tem 29 bits e identifica a mensagem (prioridade, evento)
36 //o bloco de dados deve possuir até 8 bytes
37
38 Serial.println("Enviando pacote estendido...");
39
40 CAN.beginExtendedPacket(0xabcdef); //id 11259375 decimal ( abcdef em hexa) = 24 bits preenchidos até aqui
41 CAN.write('w'); //1º byte
42 CAN.write('o'); //2º byte
43 CAN.write('r'); //3º byte
44 CAN.write('l'); //4º byte
45 CAN.write('d'); //5º byte
46 CAN.endPacket(); //encerra o pacote para envio
47
48 Serial.println("Enviado.");
49
50 delay(1000);
51 }
```

Enviando pacote...

Enviado.

Enviando pacote estendido...

Enviado.

Código-fonte do Receptor

Receptor

Código-fonte: Includes e Setup()

CANReceiver.ino ✕

```
1  #include <CAN.h> //Inclui a biblioteca CAN
2
3  void setup() {
4      Serial.begin(9600); //inicia a serial para debug
5      while (!Serial);
6
7      Serial.println("Receptor CAN");
8
9      // Inicia o barramento CAN a 500 kbps
10     if (!CAN.begin(500E3)) {
11         Serial.println("Falha ao iniciar o controlador CAN"); //caso não seja possível iniciar o controlador
12         while (1);
13     }
14 }
```

Receptor

Código-fonte: Loop(), obtendo o pacote e verificando o formato

```
15
16 void loop() {
17     // Tenta verificar o tamanho do pacote recebido
18     int packetSize = CAN.parsePacket();
19
20     if (packetSize) {
21         // Se temos um pacote
22         Serial.println("Recebido pacote. ");
23
24         if (CAN.packetExtended()) { //verifica se o pacote é estendido
25             Serial.println("Estendido");
26         }
27     }
```

Receptor

Código-fonte: Loop(), verifica se é um pacote remoto

```
if (CAN.packetRtr()) {  
    //Verifica se o pacote é um pacote remoto (Requisição de dados), neste caso não há dados  
    Serial.print("RTR ");  
}
```

Receptor

Código-fonte: Loop(), comprimento do dado solicitado ou recebido

```
33 Serial.print("Pacote com id 0x");
34 Serial.print(CAN.packetId(), HEX);
35
36 if (CAN.packetRtr()) { //se o pacote recebido é de requisição, indicamos o comprimento solicitado
37     Serial.print(" e requisitou o comprimento ");
38     Serial.println(CAN.packetDlc()); //obtem o DLC (Data Length Code, que indica o comprimento dos dados)
39 } else {
40     Serial.print(" e comprimento "); // aqui somente indica o comprimento recebido
41     Serial.println(packetSize);
42 }
```


Receptor

Código-fonte: Loop(), se há dados recebidos, os imprime

```
43 //Imprime os dados somente se o pacote recebido não foi de requisição
44 while (CAN.available()) {
45     Serial.print((char)CAN.read());
46 }
47 Serial.println();
48 }
49
50 Serial.println();
51 }
52 }
```

Recebido pacote.

Pacote com id 0x12 e comprimento 5

hello

Recebido pacote.

Estendido

Pacote com id 0xABCDEF e comprimento 5

world

Em www.fernandok.com

Download arquivos PDF e **INO** do código fonte

