|  |  |
| --- | --- |
| |  | | --- | | **УНИВЕРЗИТЕТ У НОВОМ САДУ**  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  **НОВИ САД**  **Департман за рачунарство и аутоматику**  **Одсек за рачунарску технику и рачунарске комуникације**  **ЗАВРШНИ (BACHELOR) РАД**  **Кандидат: Филип Јашић**  **Број индекса: РА46/2014**  **Тема рада: TODO**  **Ментор рада: доц. др Богдан Павковић**  **Нови Сад, јул, 2019.** | |

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | | Монографска документација | |
| Тип записа, **ТЗ**: | | Текстуални штампани материјал | |
| Врста рада, **ВР**: | | Завршни (Bachelor) рад | |
| Аутор, **АУ**: | | **Филип Јашић** | |
| Ментор, **МН**: | | **доц. др Богдан Павковић** | |
| Наслов рада, **НР**: | | TODO | |
| Језик публикације, **ЈП**: | | Српски / ћирилица | |
| Језик извода, **ЈИ**: | | Српски | |
| Земља публиковања, **ЗП**: | | Република Србија | |
| Уже географско подручје, **УГП**: | | Војводина | |
| Година, **ГО**: | | **2019.** | |
| Издавач, **ИЗ**: | | Ауторски репринт | |
| Место и адреса, **МА**: | | Нови Сад; трг Доситеја Обрадовића 6 | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | **7/58/14/3/38/0/0** | |
| Научна област, **НО**: | | Електротехника и рачунарство | |
| Научна дисциплина, **НД**: | | Рачунарска техника | |
| Предметна одредница/Кључне речи, **ПО**: | |  | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | | У библиотеци Факултета техничких наука, Нови Сад | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | | TODO | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: | проф. др |  |
|  | Члан: | доц. др | Потпис ментора |
|  | Члан, ментор: | доц. др |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual printed material | |
| Contents code, **CC**: | | Bachelor Thesis | |
| Author, **AU**: | | **Filip Jašić** | |
| Mentor, **MN**: | | **Bogdan Pavkovic, PhD** | |
| Title, **TI**: | | TODO | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian | |
| Country of publication, **CP**: | | Republic of Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | | **2019.** | |
| Publisher, **PB**: | | Author’s reprint | |
| Publication place, **PP**: | | Novi Sad, Dositeja Obradovica sq. 6 | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | **7/48/14/0/12/0/0** | |
| Scientific field, **SF**: | | Electrical Engineering | |
| Scientific discipline, **SD**: | | Computer Engineering, Engineering of Computer Based Systems | |
| Subject/Key words, **S**/**KW**: | |  | |
| **UC** | |  | |
| Holding data, **HD**: | | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | TODO | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: | , PhD |  |
|  | Member: | ,PhD | Menthor's sign |
|  | Member, Mentor: | ,PhD |  |

**Захвалност**

**Садржај**

[1. Увод 8](#_Toc14087698)

[2. Теоријске основе 10](#_Toc14087699)

[2.1 *CAN* 10](#_Toc14087700)

[2.1.1 Физички слој *CAN* – а 11](#_Toc14087701)

[2.1.2 Електрична својства, каблови и утичнице 13](#_Toc14087702)

[2.1.3 Порука, формат оквира и пренос података 14](#_Toc14087703)

[2.1.4 Арбитража 18](#_Toc14087704)

[2.1.5 Убацивање бита, препознавање грешака, сигурност 19](#_Toc14087705)

[2.2 *LIN* 21](#_Toc14087706)

[2.2.1 Топологија и физички слој 22](#_Toc14087707)

[2.2.2 *LIN* оквир 23](#_Toc14087708)

[2.3 *SPI* 26](#_Toc14087709)

[2.3.1 Пренос података, фаза и поларитет 27](#_Toc14087710)

[2.4 *I²C* 30](#_Toc14087711)

[2.4.1 Начин функционисања 30](#_Toc14087712)

[2.5 Упоредно поређење протокола 32](#_Toc14087713)

[3. Концепт решења 33](#_Toc14087714)

[3.1 Осцилоскоп 33](#_Toc14087715)

[3.2 Повезивање *CAN* – и мерење са осцилоскопом 34](#_Toc14087716)

[3.3 Повезивање *LIN* – а и мерење са осцилоскопом 35](#_Toc14087717)

[3.4 Повезивање *SPI* – и мерење са осцилоскопом 38](#_Toc14087718)

[3.5 Повезивање *I2C –* и мерење са осцилоскопом 40](#_Toc14087719)

[4. Програмско решење 42](#_Toc14087720)

[4.1 Комуникација са осцилоскопом, мерење и обрада сигнала 42](#_Toc14087721)

[4.2 Декодовање *CAN*-а 45](#_Toc14087722)

[4.3 Декодовање *LIN* – а 46](#_Toc14087723)

[4.1 Декодовање *SPI*-а 48](#_Toc14087724)

[4.2 Декодовање *I2C*-а 49](#_Toc14087725)

[5. Тестирање и резултати 50](#_Toc14087726)

[6. Закључак 52](#_Toc14087727)

[7. Литература 53](#_Toc14087728)

**Списак слика**

[Слика 2.1 Брза CAN мрежа. 11](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089270)

[Слика 2.2 Доминантни и рецесивни бит *CAN* мреже. 12](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089271)

[Слика 2.3 Омеђавање линија за брзу *CAN* линију. 13](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089272)

[Слика 2.4 Омеђавање линија за спору *CAN* линију. 13](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089273)

[Слика 2.5 Утичница *DE-9.* 14](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089274)

[Слика 2.6 Илустративни пример *CAN 2.0 A* оквира без уметања бита и његова корелација са напоном на *CAN-H* и *CAN-L.* 15](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089275)

[Слика 2.7 Илустративно објашњење како фунционише *DLC.* 16](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089276)

[Слика 2.8 Илустративни приказ *LIN* оквира за подаке. 17](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089277)

[Слика 2.9 Пример арбитраже. 18](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089278)

[Слика 2.10 Пример убацивања бита. 19](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089279)

[Слика 2.11 Топологија типичне *LIN* мреже. 22](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089280)

[Слика 2.12 Илустрација мапирања идентификатора и парности унутар поља за заштићени идентификатор. 24](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089281)

[Слика 2.13 Пример *LIN* оквира. 25](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089282)

[Слика 2.14 Пример повезивања *SPI* руководиоца и подређеног уређаја. 27](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089283)

[Слика 2.15 Пример преноса података између *SPI* руководиоца и подређеног. 28](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089284)

[Слика 2.16 Режими рада *SPI.* 28](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089285)

[Слика 2.17 Пример фазе и поларитета за *SPI.* 29](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089286)

[Слика 2.18 Временски дијаграм *I²C.* 31](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089287)

[Слика 3.1 *Tektronix DPO 4104B.* 33](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089288)

[Слика 3.2 Приказ екрана осцилоскопа приликом снимања *CAN* сигнала 35](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089289)

[Слика 3.3 Дијаграм повезивања *CAN.* 35](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089290)

[Слика 3.4 *Raspberry Pi 2* модел Б, преглед пинова. 36](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089291)

[Слика 3.5 Илустровани приказ главне утичнице за *Carberry* модул. 36](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089292)

[Слика 3.6 Приказ екрана осцилоскопа приликом снимања *LIN* сигнала. 37](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089293)

[Слика 3.7 Дијаграм повезивања *LIN*-a. 38](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089294)

[Слика 3.8 Приказ екрана осцилоскопа приликом снимања *SPI* сигнала. 39](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089295)

[Слика 3.9 Дијаграм повезивања *SPI.* 39](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089296)

[Слика 3.10 Приказ екрана осцилоскопа приликом снимања *I2C* сигнала. 40](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089297)

[Слика 3.11 Дијаграм повезивања *I2C.* 41](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089298)

[Слика 4.1 Приказ *Simulation* *Setup* прозора унутар *CANoe*-а. 43](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089299)

[Слика 4.2 Приказ *I-Generator* прозора. 44](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089300)

[Слика 4.3 Приказ *Canoe CANdb++ Editor*-а. 44](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089301)

[Слика 4.4 Приказ ухваћеног необрађеног CAN сигнала са осцилоскопа. 45](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089302)

[Слика 4.5 Приказ ухваћеног *CAN* сигнала са осцилоскопа након обраде. 46](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089303)

[Слика 4.6 Приказ необрађеног *LIN* сигнала. 47](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089304)

[Слика 4.7 Приказ обрађеног *LIN* сигнала. 47](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089305)

[Слика 4.8 Пример обрађеног *SPI* 48](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089306)

[Слика 4.9 Приказ обрађеног *I2C* сигнала. 49](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089307)

[Слика 4.10 Пример необрађеног *I2C* сигнала. 49](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089308)

[Слика 5.1 Пример излаза успешног декодовања. 50](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089309)

[Слика 5.2 Кашњење између *CAN* оквира. 51](file:///C:\Users\fjasic\Desktop\diplomski%20git\oscilloscope-bachelor\documentation\Filip_Jasic_RA46-2014.docx#_Toc14089310)

**Списак табела**

[Табела 1 Разлике између *CAN 2.0 A* и *CAN 2.0 B.* 17](#_Toc14087769)

[Табела 2 Упоредно поређење протокола. 32](#_Toc14087770)

[Табела 3 Опис жица главне утичнице за *Carberry* модул. 37](#_Toc14087771)

[Табела 4 *Carberry* команде за *LIN* 46](#_Toc14087772)

**Скраћенице**

CAN – **C**ontroller **A**rea **N**etwork

LIN – **L**ocal **I**nterconnect **N**etwork

SPI – **S**erial **P**eripheral **I**nterface

I²C – **I**nter-**I**ntegrated **C**ircuit

ISO – **I**nternational **O**rganization for **S**tandardization

Mb/s – **m**ega**b**yte per **s**econd

kb/s – **k**ilobytes **p**er **s**econd

CRC – **C**yclic **r**edundancy **c**heck

ECU – **E**lectronic **C**ontrol **U**nit

RTR – **R**emote **T**ransmission **R**equest

MSB– **M**ost **S**ignificant **B**it

LSB – **L**east **S**ignificant **B**it

SRR – **S**ubstitute **r**emote **r**equest

IDE – **I**dentifier **e**xtension **b**it

UART – **U**niversal **a**synchronous **r**eceiver-**t**ransmitter

ASIC – **A**pplication**-s**pecific **i**ntegrated **c**ircuit

USB – **U**niversal **S**erial **B**us

# Увод

Аутомобилска индустрија [1] је грана индустрије која се бави пројектовањем, развојем, производњом, маркетингом и продајом мотроних возила. Од самог почетка развоја аутомобила, због великих трошкова производње, јавила се потреба за стандардизацијом компоненти које се уграђују у аутомобил. Главни покретач развоја мрежних технологија у аутомобилима су били напреци остварени у електронској индустрији, нарочито у Сједињеним Америчким Државама. Због стриктније контроле штетних гасова од стране закона америчке владе, морао се направити механизам за већи степен контроле самог аутомобила. Ово се постигло уграђивањем рачунара који не само да су побољшали контролу штетних емисија, већ су повећали и перформансе аутомобила, сигурност, комфор али су и допринели смањењу трошкова производње. Током година се број електронских модула у аутомобилу усложњавао (*ABS, BCM, ECU, TCU*).

Електронска контролна јединица представља уграђени систем који управља једним или више електронским модулима. У модерним возилима може да се нађе и до 80 електронских контролних јединица. Електронска контролна јединица добија податке од сензора који се конвертују у одређену јединицу, где се тако обрађени подаци даље шаљу актуаторима који извршавају одређене функције. Понекад је потребно да ови модули међусобно комуницирају (нпр. модул мотора мора да обавести модул мењача која је тренутна брзина мотора, који опет мора даље да обавести остале модуле када се промени степен преноса). Овај пренос података треба да буде брз и поуздан. Како се временом повећавао број ових компоненти, дошло је до велике комплексности у ожичавању појединих модула. Додатно, ожичавања аутомобила су се разликовала од модела до модела, што је додатно повећавало трошкове производње.

Одговор аутомобилске индустрије на претходно описане проблеме је био да се створи центрлна мрежа у возилу. Идеја је била да модули буду само „прикачени“ на ове мреже како би могли да лакше међусобно комуницирају. Овај дизајн је био лакши за производњу, одржавање и дао је додатну флексибилност за надограђивање додатних модула без промене целокупне архитектуре. Сваки модул, који представља чвор на аутомобилској мрежи, контролише специфичну компоненту и комуницира са другим модулима по потреби коришћењем стандардних протокола.

Циљеви аутомобилске мреже и њених протокола су да:

* смање трошкове производње
* имају отпорност на екстерне сметње
* задрже функционалност и у отежаним условима
* робустност и поузданост

Иако су тренутни захтеви аутомобилске мреже за брзином смањени у односу на захтеве мрежа неких других система, због убрзаног развоја технологија и све већих перформанси аутомобила, постојаће и потреба за бржим магистралама.

Задатак овог рада јесте имплементација анализе и декодовања сигнала протокола чија је употреба честа у аутомобилској индустрији, чиме би се олакшала анализа целокупног система.

Рад се састоји из пет целина:

1. **Теоријске основе** – опис протокола и осцилоскопа који се користи за анализу и декодовање сигнала.

2. **Концепт решења** – имплементација повезивања протокола.

3. **Програмско решење** – имплементација декодовања и анализирања сигнала.

4. **Тестирање и резултати** – опис тестирања програмског решења.

5. **Закључак** – анализа правца за даљи развој програмског решења.

# Теоријске основе

У овом поглављу су дата објашњења протокола који се користе у аутомобилској индустрији са акцентом на начин повезивања и изгледом оквира сваког од сигнала протокола. Након тога је дат упоредни преглед протокола (табела 2).

## *CAN*

*CAN* [2] протокол је најкоришћенији протокол који се користи у аутомобилској индустрији. Развијен од стране *Robert Bosch GmbH* 1983. године, док је званично пуштен у употребу 1986. године на *Society of Automotive Engineers (SAE)* конферецији у Детроиту. *Bosch* је објавио више верзија CAN спецификације, а најновија је *CAN 2.0*, објављена 1991. године; декодовање *CAN* сигнала у овом раду се базира на поменутој верзији.

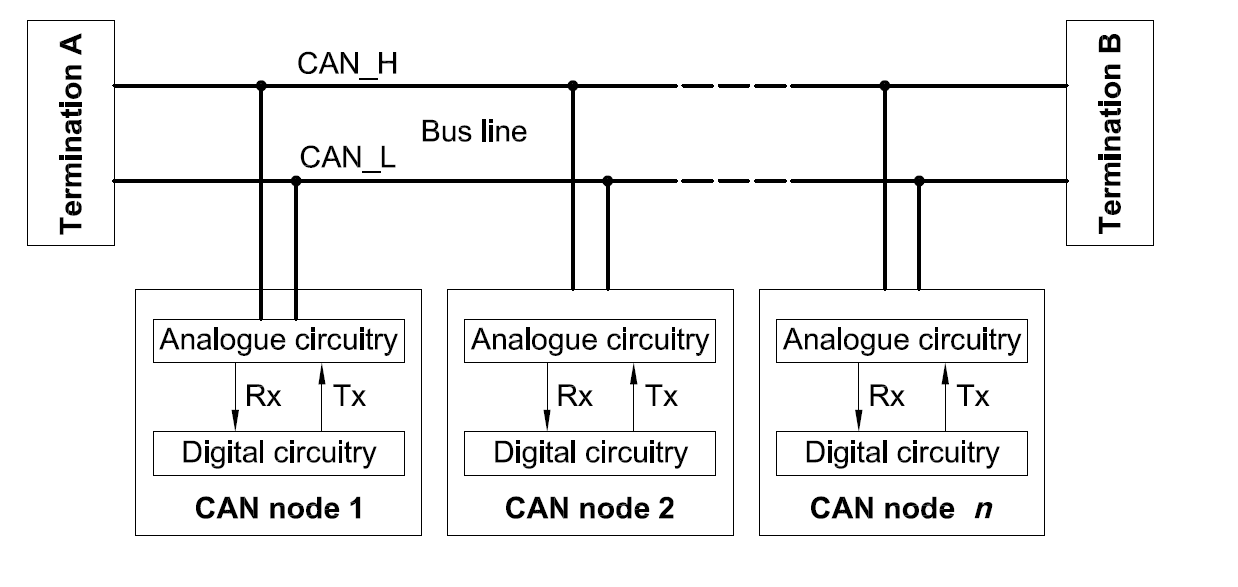
Верзија спецификације се састоји из два дела:

* А (покривен *ISO* 11519 стандардом) – овај део спецификације описује стандардни формат са 11-битним идентификатором. Мрежа где уређај користи овакав стандардни идентификатор често узима назив *CAN* *2.0 А*, коришћен надаље у раду.
* Б (покривен *ISO* 11898 стандардом) – овај део спецификације описује проширени формат са 29-битним идентификатором. Мрежа где уређај користи овај продужени идентификатор се често назива и *CAN 2.0 В* – назив коришћен надаље у раду.

Неке од карактеристика *CAN* протокола [3] су:

* Брза, серијска, заједничка магистрала (брзине до *1Mbps*) за више уређаја (до 32 уређаја по *ISO* 11898; слободан приступ – било ко може да приступа када медијум није заузет), коришћење упредене парице (домет до 40m);
* Асинхрона комуникација (окидање на догађаје);
* Величина оквира: 0-8 октета;
* Филтрирање на пријему, препознавање грешке (коришћењем цикличне провере редудансе), мере за ограничавање грешака;
* Одзив у реалном времену;
* Недеструктивна арбитража (100% искоришћеност пропусне моћи);
* Приоритет на основу идентификатора;
* Већа кашњења за поруке ниског приоритета;
* Мала кашњења за поруке високог приоритета.

### Физички слој *CAN* – а



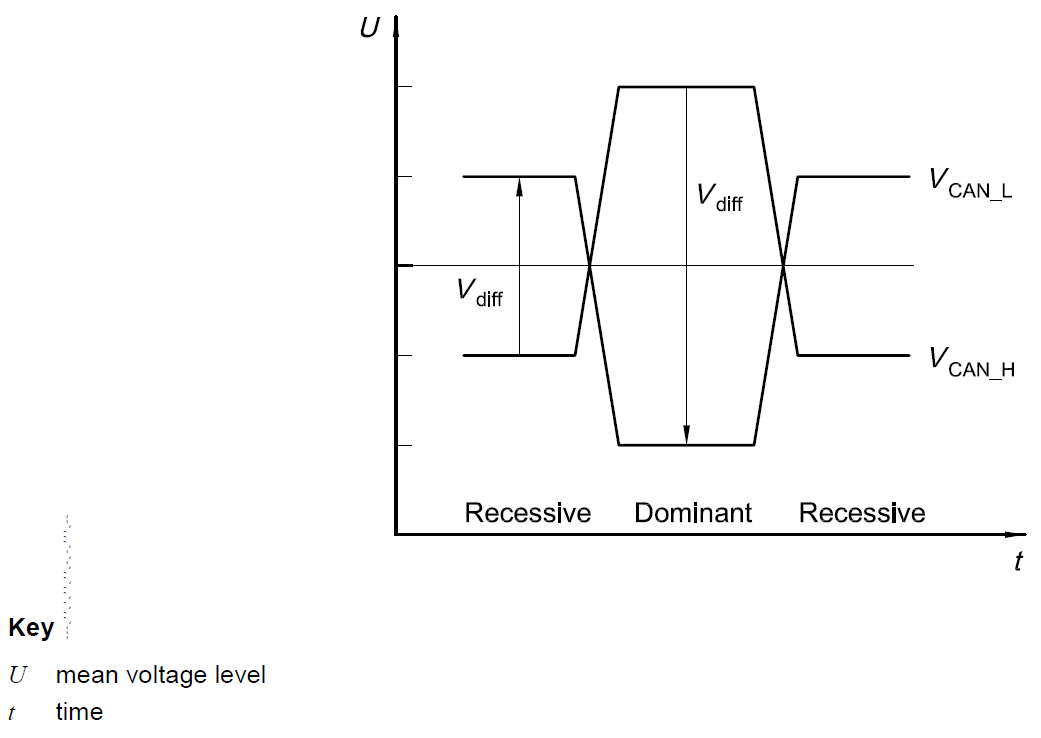
Слика 2.1 Брза CAN мрежа.

Слика 2.1 Брза *CAN* мрежа.

*CAN* мрежа поседује могућност да има више власника (енгл. *masters*). Све електронске управљачке јединице се међусобно повезују на магистралу са две жице (једна се зове *CAN-H*, друга *CAN-L*). Ове две жице чине упредену парицу (оклопљена или неоклопљена) и потребно их је омеђити отпорником са номиналном карактеристичном импеданцом од 120 Ω у случају брзе *CAN* магистрале, док у случају споре *CAN* магистрале се користи отпорник од бар 100 Ω (отпорност се засебно израчунава).

Логичка стања се процењују на основу разлике напона. Напонски нивои зависе од конкретног сучеља:

* За брзу *CAN* магистралу (*ISO* 11898-2) [4] – слика 2.1 :
  + Логичка 1 представља разлику од 0 V (у теорији) између *CAN-H* и *CAN-L* жице; у пракси се та разлика креће од 0.5 V. Напон на *CAN-H* жици се креће ка 5 V, док се на *CAN-L* жици креће ка 0 V. Логичка 1 представља рецесивни бит (високо стање).
  + Логичка 0 представља разлику од 2 V (мора биту у распону између 1.5 V и 3.5 V, а разлика напона иде чак и преко 0.9 V уколико је напајање +/-12 V) између *CAN-H* и *CAN-L* жице. Напон *CAN-H* жице иде ка 5 V док напон *CAN-L* жице иде ка 0 V. Логичка 0 представља доминантни бит (ниско стање). Представљање доминатног бита нулом се ради да би електронске контрлне јединице са најнижим идентификационим бројем имали највећи приоритет ( слика 2.2 ).
* За спору *CAN* магистралу (*ISO* 11898-3):
  + Логичка 1 представља разлику од бар 0.6 V између *CAN-H* и *CAN-L* жице. Напон *CAN-H* жице иде ка 5 V, док напон жице *CAN-L* иде при преносу логичке 0, која представља рецесивни бит.
  + Логичка 0 представља разлику од бар 2.3 V између *CAN-H* и *CAN-L* жице. Напон *CAN-H* жице иде ка 5 V, док напон жице *CAN-L* иде при преносу логичке 0, и представља доминантни бит.

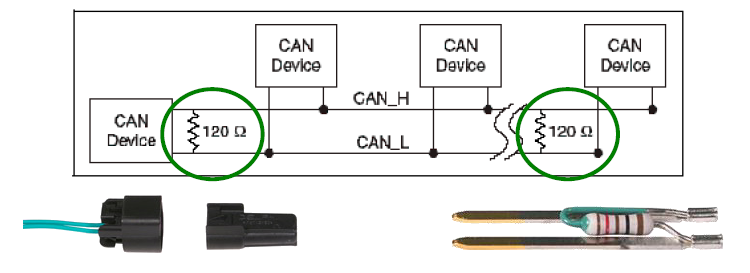


Слика . Доминантни и рецесивни бит *CAN* мреже.

Слика 2.2 Физичка репрезентација бита у брзој CAN мрежи,ISO 11898-2

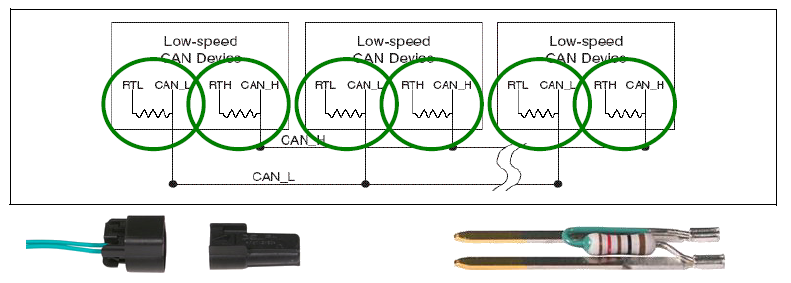
Разлике у ограђивању брзе и споре CAN магистрале су следеће:

* За брзу *CAN* магистралу је потребно да се оба краја две линије (*CAN-H* и *CAN-L*) ограде, док је средина слободна. Начин повезивања за овај тип магистале је илустрован на слици 2.3.
* За спору *CAN* магистралу нема потребе за ограђивањем крајева линије. Омеђавају се уређаји и свака линија посебно: *RTH-CAN-H* и *RTH-CAN-L*. Начин повезивања за овај тип магистрале је илустрован на слици 2.4.

****

Слика . Омеђавање линија за брзу *CAN* линију.

Слика 2.3 Омеђавање линије за брзу CAN линију

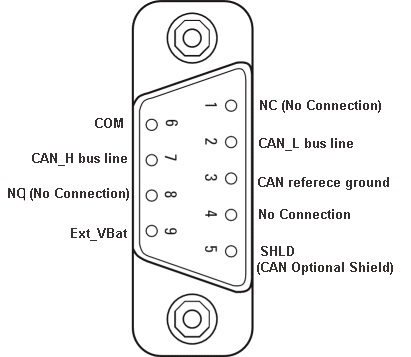


Слика . Омеђавање линија за спору *CAN* линију.

### Електрична својства, каблови и утичнице

Брзина транзиције је већа када се деси промена из рецесивног у доминантни бит, без обзира да ли је у питању брза или спора *CAN* мрежа. Брзина транзиције из доминантног у рецесивни бит зависи од дужине саме *CAN* мреже и капацитивности коришћене жице. Спецификације захтевају да магистрала буде унутар минималног и максималног заједничког напона, али не дефинишу саме вредности тих граница.

Најчешће се за каблове користе оклопљене упредене парице. Користи се један кабел, осим у случају када постоји потреба за додатним напајањем. Сам *CAN* не одређује врсту физичког медијума, међутим, типично се користе 9-пинске *D-sub,* и   
5-пинске *mini style* утичице. Пример утичнице која се најчешће користи у аутомобилској индустрији са *CAN* протоколом (*DE-9* тип *D-sub* утичнице) је приказан на слици 2.5 .



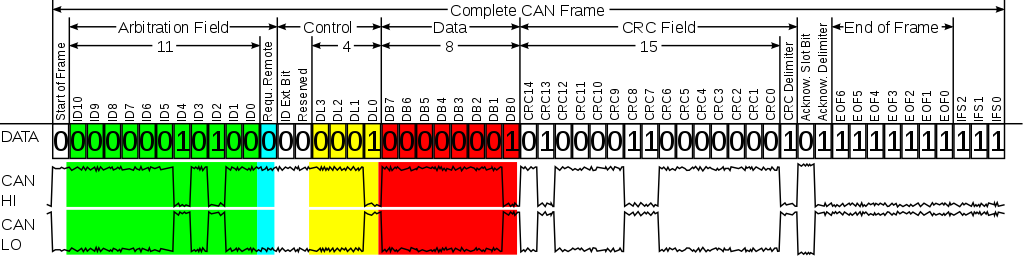
Слика . Утичница *DE-9.*

### Порука, формат оквира и пренос података

Уређаји повезани на *CAN* комуницирају слањем порука. Пренос порука је контролисан са четири типа оквира:

* Оквир за податке (енгл. *Data frame*) – носи податке од предајника до пријемника.
* Оквир за даљинско управљање (енгл. *Remote frame*) се шаље на магистралу као захтев за оквиром за податке са одређеним идентификатором.
* Оквир за грешке (енгл. *Error frame*) се шаље од било ког чвора уређаја који детектује грешку на магистрали.
* Оквир за пренатрпавање (енгл. *Overload frame*) се користи да омогући кашњење између два послата оквира за податке или оквира за грешке.

Илустративни пример *CAN 2.0 A* без уметања бита је приказан на .



Слика . Илустративни пример *CAN 2.0 A* оквира без уметања бита и његова корелација са напоном на *CAN-H* и *CAN-L.*

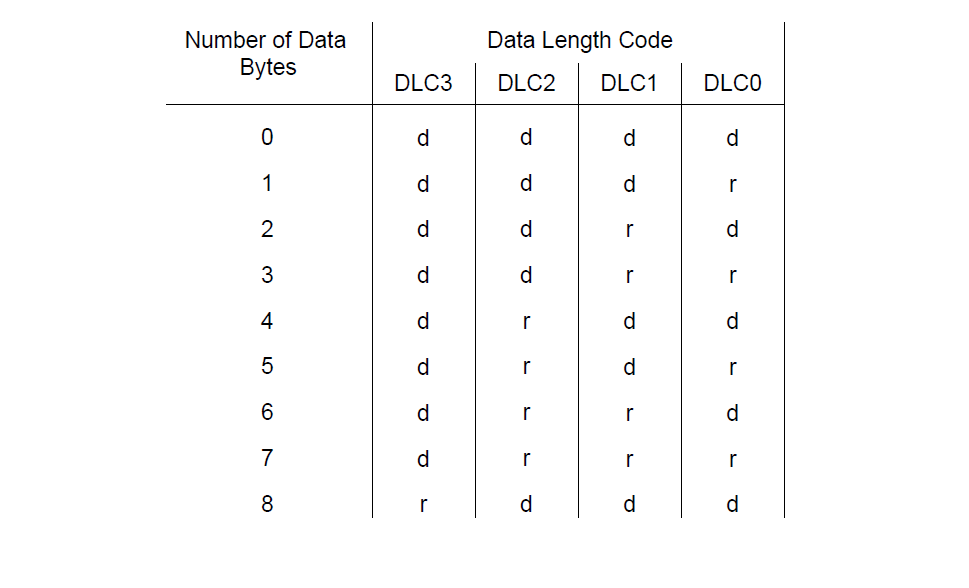
Оквир за податке се састоји од седам поља (разлике између *CAN 2.0* *A* и *CAN 2.0 B* су по потреби посебно наведени, док се подразумева да одређена поља имају исту структуру):

* Почетак оквира (енгл. *Start of frame*) означава почетак поља за оквир за податке или оквира за даљинско управљање. Састоји се од једног доминантног бита (логичка 0). Уређај на магистрали може да почне да шаље када је магистрала у мирном режиму рада (енгл. *Bus Idle*). Сви уређаји морају да се синхронизују са водећом ивицом која је проузрокована од стране овог поља од уређаја који је први започео пренос.
* Поље за арбитражу (енгл. *Arbitration field*). За ово поље постоје разлике између *CAN 2.0 A* и *CAN 2.0 B*:
  + Ово поље се састоји од идентификатора и *RTR* бита у случају *CAN 2.0 A,* при чему идентификатор се састоји од 11 бита; ови бити се гледају у *MSB* редоследу при чему свих 7 најзначајних бита (од *ID-10* до *ID-4*) не смеју бити рецесивни. *RTR* бит мора да буде доминантан када се шаље оквир за податке, иначе, када се шаље оквир за даљинско управљање овај је бит рецесиван.
  + У случају *CAN 2.0 B* идентификатор се састоји од 29 бита, подељених на два дела: идентификатор А (11 бита) и идентификатор Б(18 бита). Између ова два дела продуженог идентификатора се налази *SRR* (1 рецесиван бит) и *IDE* (1 рецесиван бит).
* Контролно поље (енгл. *Control field*). За ово поље такође постоје разлике између две подврсте протокола:
  + Код бржег типа протокола, контролно поље се састоји од 6 бита. Оно укључује 4 бита који означавају дужину послатих података (енгл. *Data length code*) у бајтима, и 2 бита који су резервисани за будућа проширења протокола – *IDE* бит (мора бити доминантан) и резервисани *r0* бит (може бити или доминантан или рецесиван).
  + Код споријег типа протокола ово поље се такође састоји од 6 бита, при чему прво стижу 2 резервисана бита – *r0*, *r1* (могу бити или доминанти или рецесивни). Након њих у оквиру се налазе 14 бита који означавају дужину послатих података и функционишу на исти начин као у бржем протоколу.

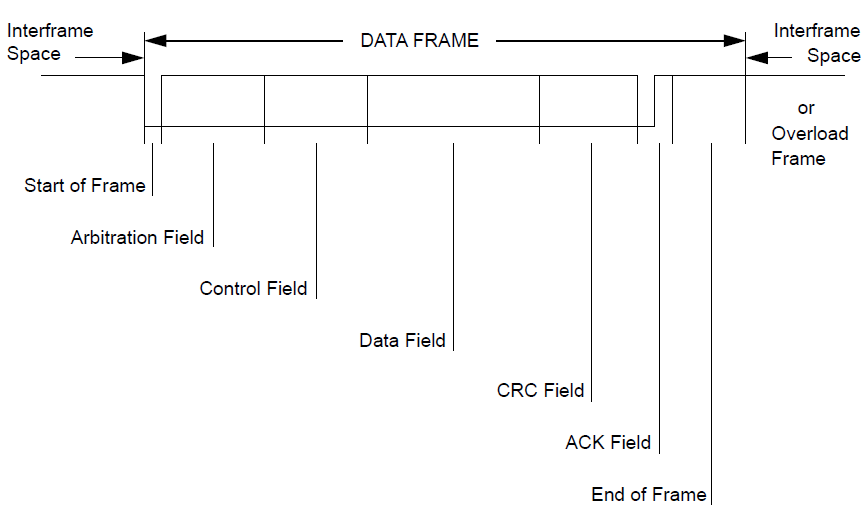
Илустративно објашњење како фунционише део котнролног поља које означава дужину података је приказано на при чему су са *d* обележени доминантни бити док су са *r* обележени рецесивни бити.

* Поље за податке (енгл. *Data field*) се састоји од 0 до 8 бајта, при чему 1 бајт има 8 бита. Подаци се шаљу у MSB редоследу.
* Поље за цикличну проверу редудансе (енгл. *CRC field*) се састоји од 15 бита који заправо представљају израчунату цикличну проверу редудансе и делимитера који заузима 1 бит, при чему је тај бит рецесиван.
* Поље за потврду о пријему (енгл. *ACK field*) се састоји од *ACK slot* и *ACK* делимитера где сваки заузима по 1 бит и при чему пошиљаоц поставља рецесивни бит у *ACK slot*, док примаоц поруке може да нареди слање доминантног бита. *ACK* делимитер „раздваја“ ово поље са крајем оквира и оно садржи 1 рецесиван бит.
* Крај оквира (*End of frame*) се састоји од 7 рецесивних бита.

Илустративни приказ оквира за податке је приказан на .



Слика . Илустративно објашњење како фунционише *DLC.*



Слика . Илустративни приказ *LIN* оквира за подаке.

Разлике у оквирима две врсте *CAN* протокола су приказане у табели 1 .

|  |  |  |  |
| --- | --- | --- | --- |
| *CAN 2.0 А* (11-битни идентификатор) | | *CAN 2.0 B* (29-битни идентификатор) | |
| Назив поља | Дужина поља | Назив поља | Дужина поља |
| Почетак оквира | 1 бит | Почетак оквира | 1 бит |
| Идентификатор | 11 bita | Идентификатор А | 11 бита |
| *RTR* | 1 бит | *SRR* | 1 бит |
| *IDE* | 1 бита | *IDE* | 1 бит |
| Резервисан бит | 1 бита | Идентификатор Б | 18 бита |
| *DLC* | 4 бита | *RTR* | 1 бит |
| Подаци | 0-64 бита | Резервисани бити | 2 бита |
| *CRC* | 15 бита | *DLC* | 4 бита |
| *CRC* делимитер | 1 бит | Подаци | 0-64 бита |
| *ACK* | 1 бит | *CRC* | 15 бита |
| *ACK* делимитер | 1 бит | *CRC* делимитер | 1 бит |
| *EOF* | 1 бит | *ACK* | 1 бит |
|  |  | *ACK* делимитер | 1 бит |
|  |  | *EOF* | 1 бит |

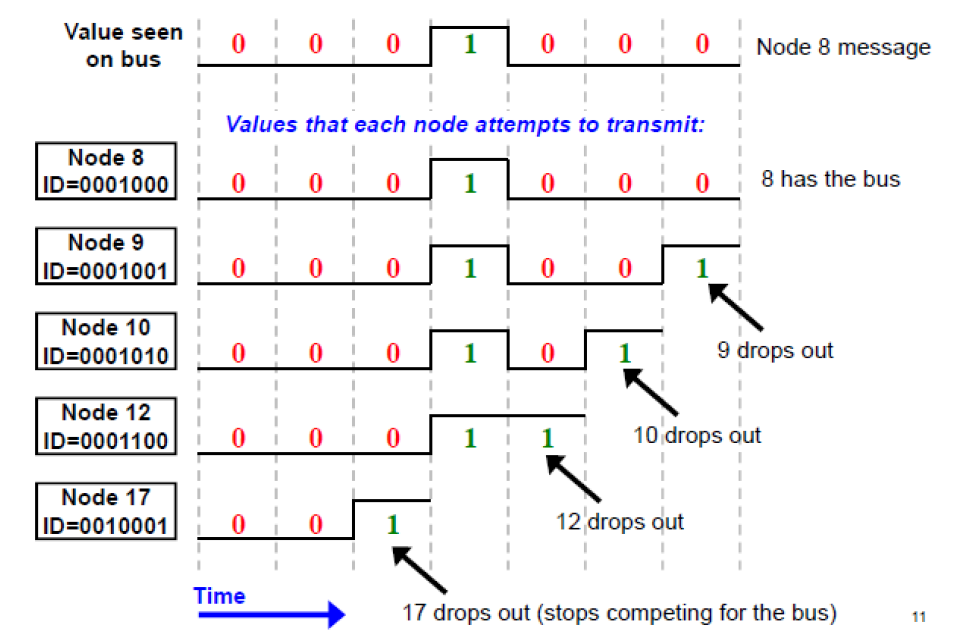
Табела 1 Разлике између *CAN 2.0 A* и *CAN 2.0 B.*

* Поље за почетак оквира
* Поље за арбитражу
* Контролно поље
* Поље за податке
* Поље за цикличну проверу редудансе
* Поље за потврду о пријему
* Крај оквира

### Арбитража

Када год је магистрала слободна, било која електронска управљачка јединица може да почне са слањем поруке. У случају да две или више ових управљачких јединица почне са слањем порука у исто време настаје конфликт који се решава механизмом арбитраже за појединачне бите коришћењем идентификатора. Овај механизам обезбеђује да се не губи на времемену, а такође спречава губитак података. Ако оквир за пренос података и оквир за даљинско управљање имају исти идентификатор и ако су иницијализовани у исто време, предност има оквир за слање података.

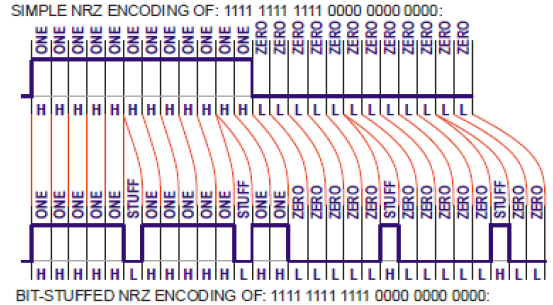
Током арбитраже, сваки пошиљаоц проверава на ком нивоу се налази бит идентификатора који је он послао поредећи га наспрам нивоа бита идентификатора који се налази на магистрали. Ако су ови бити на истим нивоима једнаки, пошиљаоц може да настави са слањем. Када је рецесивни бит на истом нивоу послат и доминантни бит на истом нивоу се налази на магистрали, тада је пошиљаоц „изгубио“ арбитражу и мора да се повуче, при чему се прекида пренос преосталих бита у оквиру. Другим речима, побеђују уређаји са најмањим идентификатором. Идентификатор се шаље у *MSB* формату, да би се брже уочила арбитража и да би медијум био ефикаснији за слање. Илустративни пример арбитраже је приказан на слици 2.9 .



Слика . Пример арбитраже.

### Убацивање бита, препознавање грешака, сигурност

Убацивање бита се врши на следећи начин: након 5 идентичних бита се убацује супротан бит; ово побољшава усклађивање а додатни бити не мењају податке пошто је убацивање урађено хардверски. Ова техника генерално није потребна, пошто се врло ретко појављују истоветне вредности на магистрали. Проблем са убацивањем бита је тај што се инверотовани бити тешко могу препознати, а проблем такође представља и појава такозваних „лавинских“ битова (нпр. 5 рецесивних бита које прати 5 доминантих бита) који могу да збуне механизам за препознавање убачених бита. Понекад циклична провера редудансе може да помогне али са додатним ограничењима. Једноставан пример убацивања бита је приказан на слици 2.10 .



Слика . Пример убацивања бита.

Статистичка учестаност грешака *CAN* протокола зависи од укупног броја уређаја, физичког ожичења и распореда, као и спољашњих електромагнетних сметњи. Учестаност непрепознатих грешака је једном у хиљаду година, и то је један од разлога зашто је ово најкоришћенији протокол у аутомобилској индустрији која захтева дуготрајну поузданост.

Ток препознавања грешака између пошиљаоца (*Tx*) и примаоца (*Rx*) је независан од филтрирања и маскирања на пријему и састоји се из пет засебних механизама:

1. Надгледање појединачних бита (*Tx*) – препознавање локалних и глобалних грешака код пошиљаоца; упоређује се стање на магистрали са послатим битом;

не примењује се на поље за арбитражу.

1. Провера структуре – типична поља су *CRC* делимитер, *ACK* делимитер и *EOF* и чине их увек рецесивни бити.
2. Провера кодовања (уметање бита) – пријемник проверава ток бита, на сваких 5 узастопних истоветних бита мора да дође убачена промена; укључује све бите од *SOF* do краја *CRC*.
3. Провера потврде (*Tx*) је обавеза пошиљаоца – потребно је бар да пристигне једна потврда па да пошиљалац подеси рецесивни бит; такође, потребно је да пријем обори са доминантним.
4. Циклична провера редудансе – пошиљалац прорачунава пре слања, пријемник проверава подударање при пријему. Циклична провера редудансе се рачуна на следећи начин: полином, чији су коефицијенти представљени комбинацијом вредности претходних поља (при чему је претходно урађено избацивање убачених бита), а коефицијенати 15 најнижих бита чине нуле. Овај полином се потом дели (коефицијенти су израчунати модулом двојке) са генератор-полиномом:

*x15 + x14 + x10 + x8 + x7 + x4 + x3 + 1*

Oстатак овог дељења представља 15-битну вредност *CRC* секвенце која се шаље на магистралу као додатак оригиналне поруке. Да би се применила ова функција, 15-битни померачки регистар *CRC\_RG* (14:0) се може користити. Псеудо-код ове функције се описује са:

CRC\_RG = 0; //иницијализација померачког регистра

REPEAT

CRCNXT = NXTBIT EXOR CRC\_RG(14); //NXTBIT означава следећи бит у низу

CRC\_RG(14:1) = CRC\_RG(13:0); //померај у лево за једно место

CRC\_RG(0) = 0; IF CRCNXT THEN

CRC\_RG(14:0) = CRC\_RG(14:0) EXOR (4599hex);

ENDIF

UNTIL (CRC SEQUENCE starts or there is an ERROR condition)

Након слања/пријема последњег бита поља за податке, *CRC\_RG* садржи *CRC* секвенцу.

Сам протокол не подржава енкрипцију у стандардној имплементацији и у већини примена, од уређаја који користе *CAN,* се очекује да имају своје сигурносне механизме. Ако ово није урађено уређаји су подобни раличитим врстама малициозних напада, нпр, ако неко пошаље своје поруке на магистралу отворен је приступ за манипулацију са уређајем. Постоје шифровани системи који су задужени за фунцкионалности од критичног значаја за сигурност. Међутим, ови системи нису универзално имплементирани.

## *LIN*

*LIN* [5] је серијски мрежни протокол, развијен од стране *LIN* конзорцијума (5 произвођача аутомобила – BMW, Volkswagen Group, Audi Group, Volvo Cars,  
Mercedes-Benz; један добављач полупроводиника – Motorola; и један добављач  
алата – VCT) основаног 1998. године, док је прва, у потпуности имплементирана верзија *LIN* спецификације (LIN 1.3) објављена 2002. године. Иницијално је развијена као *CAN* подмрежа са циљем да се смањи оптерећење, међутим, данас има широку употребу у раличитим индустријама: у аутомобилима, медицинској опреми, белој техници, итд.

Најважније одлике *LIN* прокола су:

* Брзина до 20 *kb/s* на дужинама до 40 метара.
* Руководилац контролише медијум, због чега постоји гаранција кашњења и нема судара. Усклађивање такта је одговорност зависних уређаја (због тога нема потребе за кварцним или керамичким резонаторима).
* Варијабилност у дужини послатих података (2, 4 или 8 бајта, при чему 1 бајт чини 8 бита).
* Флексибилност у конфигурацији и динамичка топологија.
* Величина мреже је углавном до 12 чворова (могуће је и до 64, али уз низак проток).
* Детекција дефектних чворова, детекција грешке и контролни збир података.
* Оперативни напон од 12 V.
* Ниска цена, силиконска имплементација заснована на *UART* стандарду.
* Предности *LIN* су: лак за коришћење, велика доступност компоненти, јефтинија него *CAN* и друге магистрале за комуникацију, возила су поузданија, продуживање магистрале је лако за имплементацију, не постоје трошкови за лиценцирање.

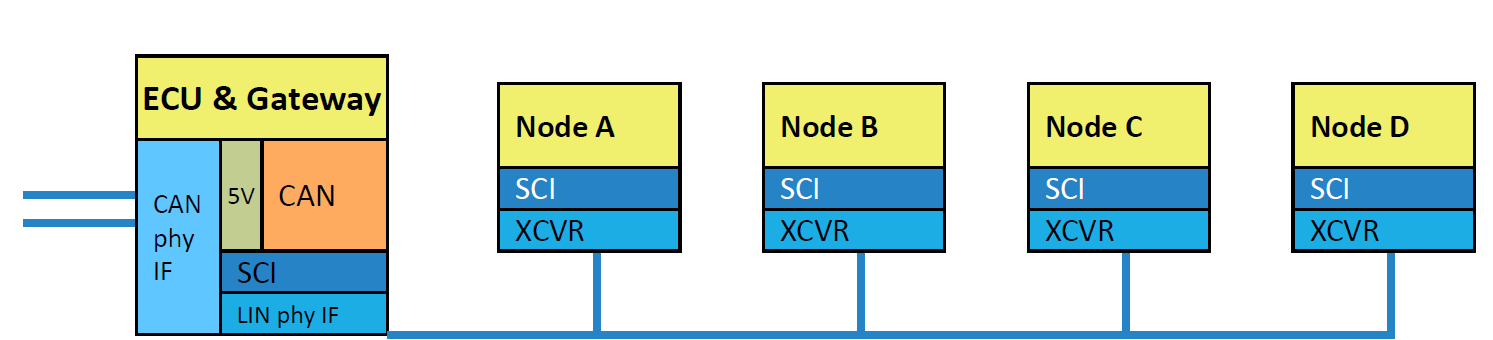
Постоји више верзија *LIN* протокола, од којих су нам две релевантне надаље у  
раду – 1.3 и 2.0. Нека од унапређења новије верзије су :

* Низови бајтова су подржани тако да омогућавају дужину сигнала до 8 бајта.
* Групе сигнала се бришу.
* Проширени контролни збир (који укључује и заштићени идентификатор) је унапређење у односу а класични контролни збир код *LIN* *1*.*3*.
* Спорадични оквири су дефинисани.
* Управљање мрежом је мерено у секундама, не у битима.
* Управљање статусом је поједностављено а обавештавање мрежи и апликацијама је стандардизовано.

### Топологија и физички слој

*LIN* [6] је мрежа са дифузним емитовањем која се састоји од најчешће 16 мрежних чворова (један руководилац и до петнаеаст подређених). Све поруке су иницијализоване од стране руководиоца са највише једним подређеним који одговара на тренутно послату поруку.

Руководилац такође може да се понаша као подређени тако што одговара на своје поруке. Пошто је целокупна комуникација иницијализована од стране руководиоца, није потребна имплементација детекције судара. Уређаји повезани на *LIN* мрежу су углавном микроконтролери, али могу бити имплементирани и у специјализованом хардверу или *ASIC* у намери да се додатно смање трошкови, површина или потребна струја. Једна типична *LIN* мрежа је приказана на слици 2.11 .



Слика . Топологија типичне *LIN* мреже.

Физички слој се представља као коло са отвореним колектором, магистала је терминирана руководиоцом (1 kOhm). Напон је између 7 V и 18 V на прикључцима. Постоје строги захтеви за симетријом и нагибима ивица, а мрежа је осетљива због велике толеранције осцилатора. Радни циклус има минимум од 39.6% а максимум од 58.1%.

### *LIN* оквир

Време које је потребно да се пренесе *LIN* оквир на магистралу је сума свих времена за које је потребно да се пренесе сваки бајт, времена за одговор и времена између два оквира. Време између 2 бајта је период између краја једног бита и времена почетка наредног бита.

Структура оквира се састоји од следећих делова:

* Заглавље – унутар кога се налази:
  + Поље за почетак (енгл. *Break*) се користи за сигнализацију почетка оквира. Оно се увек генерише од стране главног, управљачког чвора (енгл. *master node*) и мора да има величину од бар 13 доминантних бита, укључујући почетни бит и након њих, делимитер за почетак поља.
  + Поље за синхронизацију (енгл. *Synch*) увек има вредност која у хексадецималном формату износи 0x55, да би уређаји на мрежи увек могли да препознају ово поље. Уређаји којима се управља (енгл. slaves) увек имају могућност да детектују секвенцу поља за почетак и овог поља, чак иако очекују поље за податке.
  + Заштићени идентификатор (енгл. *Protected identifier*) се састоји од пет под-поља: идентификатор и парност идентификатора. Вредност од 0. до 5. бита означава идентификатор док од 6. до 7. бита је вредност парности идентификатора.

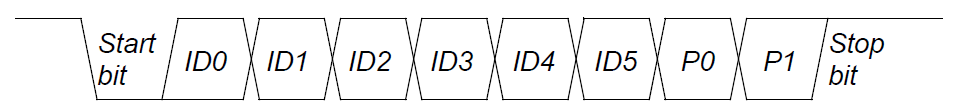
Вредности идентификатора имају и функцију одређивања примене послате *LIN* поруке па тако:

* + - 1. Вредности од 0 до 59 (*0x3b* у хексадецималном облику) се користе за пренос података.
      2. Вредности између 60 (*0x3c*) и 61 (0x3d) се користе за слање дијагностичких података.
      3. Вредност 62 (*0x3e*) је резервисана са употребе дефинисане од стране корисника.
      4. 63 (*0x3f*) је резервисано за будућа побољшања.

Парност бита се израчунава на следећи начин (при чему су вредности у једначинама објашњене у илустрацији мапирања овог поља на слици 2.12 ):

P0 = ID0 ⊕ ID1 ⊕ ID2 ⊕ ID4 (1)

P1 = ¬ (ID1 ⊕ ID3 ⊕ ID4 ⊕ ID5)



Слика . Илустрација мапирања идентификатора и парности унутар поља за заштићени идентификатор.

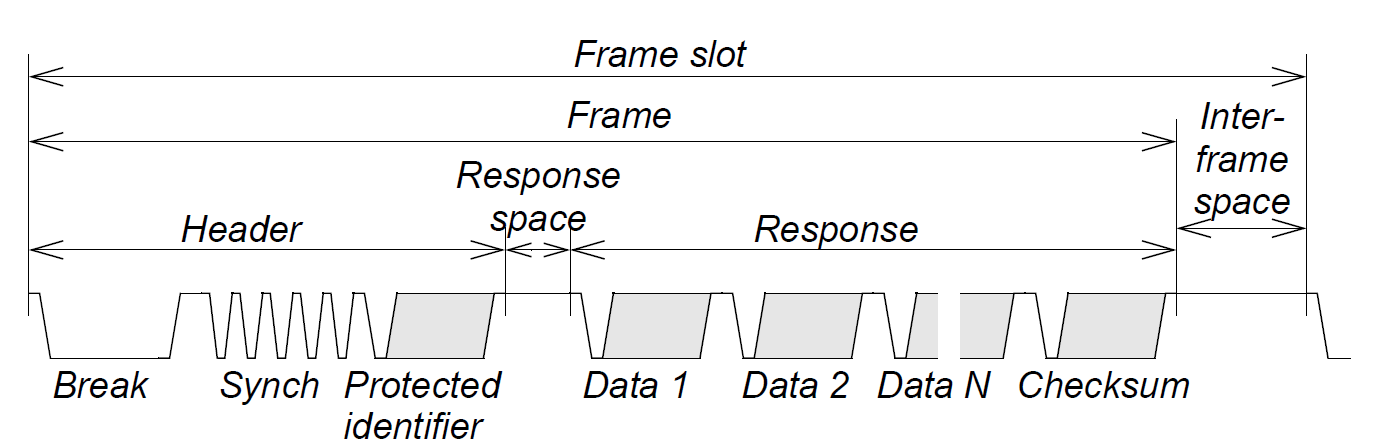
* Размак за одговор (енгл. *Response space*).
* Одговор (енгл. *Response*) – унутар кога се налазе подаци и поље за проверу контролног збира. Поље за податке се састоји од 1 до 8 бајта. Након ових поља за податке долази поље за проверу контролног збира. Оно садржи 8-битну инвертовану суму са преносом за све податке унутар поруке (у случају *LIN 1.3* – и тада се оно назива класично поље за проверу контролног збира, енгл. *classic checksum*) или инвертовану суму са преносом за све податке и заштићени идентификатор унутар поруке (такозвано побољшано поље за проверу контролног збира, енгл. *enhanced checksum*, користи се у верзији 2.0). Идентификатори са вредношћу 60 и 63 ће увек користити класичну контролну суму.

Врсте одговора су:

1. Безусловни оквир (енгл. *Unconditional Frames*) (идентфикатор 0-59) – тачно један дефинисани подређени шаље одговор, сваки захтев се шаље у посебном пакету. Потребно је обезбедити довољно велики пакет за одговор. Пакет се углавном шаље сваког циклуса распореда. Могуће је слати више пута у току истог циклуса, што то зависи од апликације. Пријем одговора је доступан свим подређенима и руководиоцу.
2. Дијагностички оквир (енгл. *Diagnostic Frames*) (идентификатор 60-61). Могуће је слати сегментирано, односно више оквира заредом без заглавља. Постоје две врсте овог оквира:
   1. Захтев (енгл. *Master Request Frame*)(идентификатор 60) – заглавље и одговор шаље руководилац. Употребљава се кад је потребан захтев за дијагностиком или подешавањем.
   2. Одговор (енгл. *Slave Response Frame*) (идентификатор 61) – заглавље шаље руководилац а одговор одговарајући подређени. Употребљава се као одговор на захтев за дијагностиком или подешавањем.
3. Временски диригован оквир (енгл. *Event triggered frame*) (идентификатор 0-59). Служи за груписање више повремених безусловних оквира. Овим се добија на уштеди у циклусу – кратики одговори који не заузимају распоред. Последица овога је да долази до ретких појава судара, који се додатно избегавају добром архитектуром мреже. Сударе решава руководилац.
4. Спорадични оквир (енгл. *Sporadic frame*) (идентификатор  
   0-59). Логика овог оквира је слична претходном оквиру.
5. Остали оквири (енгл. *Other frames*) (идентификатор 62-63).

* Размак између оквира.

Илустративни приказ оквира је приказан на слици 2.13 .



Слика . Пример *LIN* оквира.

Слика 3.2 Пример LIN оквира

Сваки оквир чије се слање планира алоцира одређени временски интервал на магистрали. Трајање овог временског интервала мора бити довољно дугачко да омогући пренос оквира чак и у најгорем могућем случају (нпр. због лошег квалитета подређених компоненти се стварају кашњења). Номинална вредност за пренос оквира је једнака броју послатих података у битима. Рачунање овог временског интервала се израчунава на следећи начин:

**Тноминално\_време\_заглавља = 34 \* Tбита**

**Тноминално\_време\_одговора = 10 \* (Број\_бајта\_података + 1) \* Тбита**

**Тноминално\_време\_оквира = Тноминално\_време\_заглавља + Тноминално\_време\_одговора**

**Тбит = 1 / брзина такта**

**Тбита** представља време потребно за пренос једног бита. Максимално време оквира је још 40% у односу на **Тноминално\_време\_оквира**.Такт је до 20 kb/s.

## *SPI*

*SPI* [7] је синхрони, серијски комуникациони интерфејс који се користи за комуникацју на кратким дистанцама, првенствено у уграђеним системима. Интерфејс је осмишљен од стране *Motorola*-е средином 80-тих година 20. века. Неке од карактеристика овог интерфејса су:

* Једноставност повезивања: са 4 жице – *SCLK* (*Serial Clock*) – такт за усклађивање преноса; *MOSI* (*Master Output Slave Input*) – преноси податке од руководиоца до подређеног; *MISO* (*Master Input Slave Output*) – преноси податке од подређеног до руководиоца; *SS* (*Slave Select*) – посебна линија за одабир сваког подређеног, један надређени и више подређених. Надређени активира подређене ниским сигналом. Обе, *MISO* и *MOSI*, линије су активне током сваког преноса.
* Брже од *UART*-а: 250 *kb/s* до 2 *Mb/s*.
* Кратак домет: ~10 cm на сваки чвор.
* Могућност повезивања процесора и широког спектра подржаних периферија, два процесора. Могуће је и повезивање више зависних подређених уређаја (један руководилац и више нанизаних подређених – енгл. *Daisy chained*).
* Групе бита података (4, 8, 16); непостојање стартног и стоп бита. Кодовање  
  *NRZ-L*. Одвојене линије за пренос у оба смера (дуплекс, али је потребна подршка на чипу).
* Синхронизација – посебна линија за такт (неактивна кад се постави на 1 или 0 на дужи период). Нема потребе за усклађивањем такта.
* Читање на опадајућу или растућу ивицу.

*SPI* поседује четири регистра:

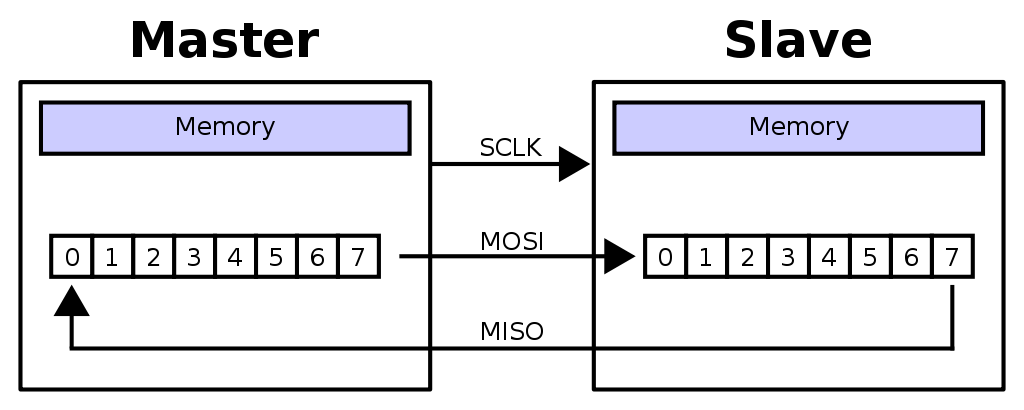
1. *SDPR* (*transferred Data read/write Register*) – смештање примо-предајних података.
2. *SPCR* (*Control Register*) – смештање контролних подешавања.
3. *SPSR* (*Status Register*) – смештање статусних информација.
4. *SPTCI* (*Transfer Complete Interrupt*) – смештање прекидачке рутине за готов пренос.

### Пренос података, фаза и поларитет

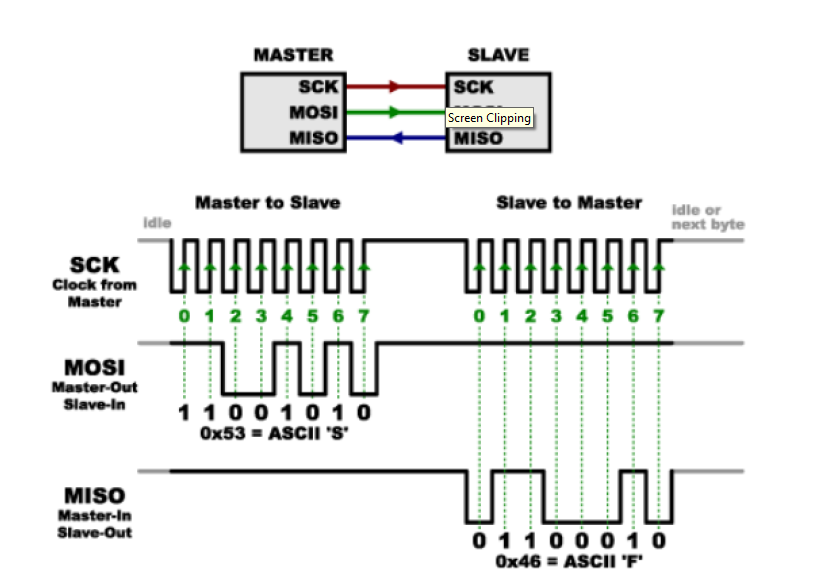
Да би започео комуникацију, руководилац магистрале прво конфигурише радни такт, користећи фреквенцију подржану од стране подређеног уређаја; углавном до пар *MHz*. Руководилац потом бира подређени уређај са логичком 0 на *SS* линији. Ако се захтева период чекања, руководилац мора да чека тај период пре него што пошаље радни такт.

Након сваке периоде радног такта, долази до преноса у оба смера – пуни дуплекс. Руководилац шаље бит на *MOSI* линији и потом га подређени прочита, док је на *MISO* линији обрнуто. Овај процес се дешава чак и када се подаци шаљу у једном смеру.

Пренос углавном укључује два померачка регистра, дужине података; један се налази код руководиоца, други код подређеног. Подаци се шаљу користећи *MSB* организацију. На ивицу такта оба уређаја померају по 1 бит из регистра и шаљу га даље свом одговарајућем уређају. На следећој ивици такта, сваки од уређаја прими бит са линије за пријем и тај бит се поставља као *LSB* померачког регистра. Након што су се сви бити померили и након што су послати, два уређаја су разменили податке. Када се трансмисија у потпуности обавила, руководилац престаје да генерише ивице такта. Ако треба да се пошаље још података, померачки регистри се опет испочетка „напуне“ битима и пренос опет може да започне. Пример повезивања и слања података између два уређаја на овакав начин је приказан на слици 2.14 и слици 2.15 . Трансмисија се углавном састоји од 8 бита.



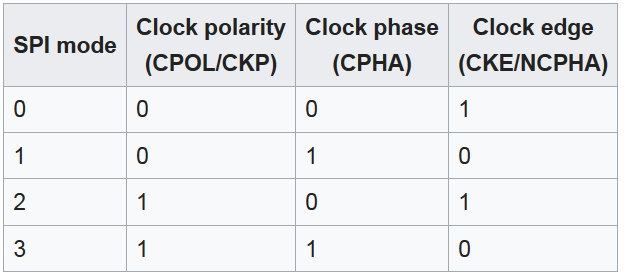
Слика . Пример повезивања *SPI* руководиоца и подређеног уређаја.



Слика . Пример преноса података између *SPI* руководиоца и подређеног.

Постоје четири режима као последица комбинација (илустровано на слици 2.16 ):

* Два стања фазе такта (енгл. *CPHA*)
* Два стања поларитета такта (енгл. *CKP/CPOL*)



Слика . Режими рада *SPI.*

\

Фаза (енгл. *CPHA*) дефинише значење предње и задње ивице такта. Пресликавање узорковања и прелаза на нови бит некој од ивица. Различите ивице имају различито значење. Поларитет (енгл. *CKP/CPOL*) дефинише вредност активног и стања мировања. Пресликавање 0 и 1 на активност и мировање. Мировање као почетак и референца на ивице.

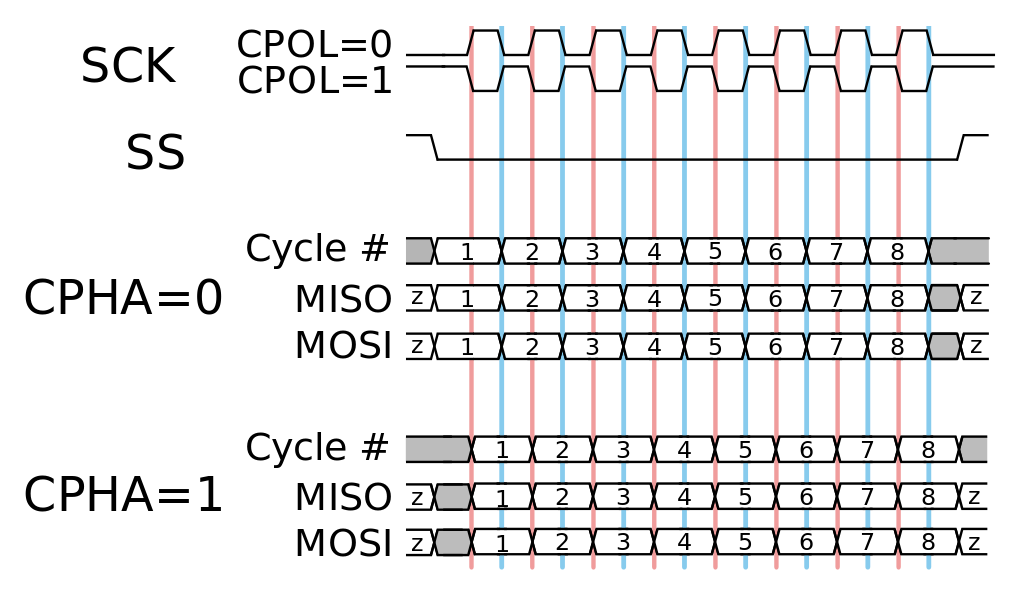
Када је *CPOL*=0 мировање је 0 а активност 1:

* За *CPHA*=0, узорковање на предњу/узлазну ивицу (0→1) а прелаз на нови бит на задњу/силазну (1→0).
* За *CPHA*=1, узорковање на задњу/силазну ивицу (1→0) а прелаз на нови бит на предњу/узлазну (0→1).

Када је *CPOL*=1 мировање је 1 а активност 0 (инверзија за *CPOL*=0):

* За *CPHA*=0, узорковање на предњу/силазну ивицу (1→0) а прелаз на нови бит на задњу/узлазну (0→1).
* За *CPHA*=1, узорковање на задњу/узлазну ивицу (0→1) а прелаз на нови бит на предњу/силазну (1→0).

*MOSI* и *MISO* сигнали су углавном стабилни за пола циклуса до следећег прелаза на нови бит. *SPI* руководилац и подређени могу да одабирају сигнале на различитим одбирцима у том полу-циклусу. Ово даје додатну флексибилност комуникационом каналу између руководиоца и подређеног. Пример фунционисања фаза и поларитета је дат на слици 2.17 .



Слика . Пример фазе и поларитета за *SPI.*

## *I²C*

*I²C* [8] је синхрона магистрала са могућношћу да има више руководиоца и више подређених створена од *Philips Semiconductor* 1982. године. Има широку употребу за повезивање спорих периферија за процесоре и микроконтроле на малим дистанцама. Од октобра 2006, не постоје трошкови лиценцирања за употребу овог протокола. *SMBus* је подскуп овог протокола, створеног од стране *Intel*-а 1995. године као одговор на потребу за дефинисаном стриктнијом употребом и циљем да се дода робустност и интер-опребилност.

Модерни *I2C* системи подржавају нека правила из *SMBus*- a, при чему је потреба за реконфигурацију мала. Оригинално употребљиван у телевизорима, сада подржава велике количине периферија (наменски системи – *EEPROM*, *Flash* и *RAM* меморија, *watchdog* тајмери, микроконтролери, персонални рачунари). *I²C* подржава више брзина комуникације:

* Стандардни режим (енгл. *Standard* *Mode*): 100 *kb/s*, оригинално од прве верзије у употреби.
* Брзи режим (енгл. *Fast Mode*): 400 *kb/s*, додат у верзији 1 овог протокола 1992. Године; додато је и 10-битно адресирање, због чега се максималан број мрежих чворова повећао на 1008. Ово је прва стандардизована верзија.
* Брзи режим плус (енгл. *Fast Mode plus*): *1 Mb/s*, додат у верзији 3 2007. године.
* Режим велике брзине (енгл. *High Speed Mode*): 3.4 *Mb/s* додат у верзији 2 1998. године.
* Ултра-брзи режим (енгл. *Ultra Fast Mode*): 5 *Mb/s* додат у верзији 4 2012. године.

### Начин функционисања

*I²C* користи две двосмерне линије – *SDA* (*serial data*) и *SCL* (*serial clock*). Имплементација преко отвореног колектора омогућава:

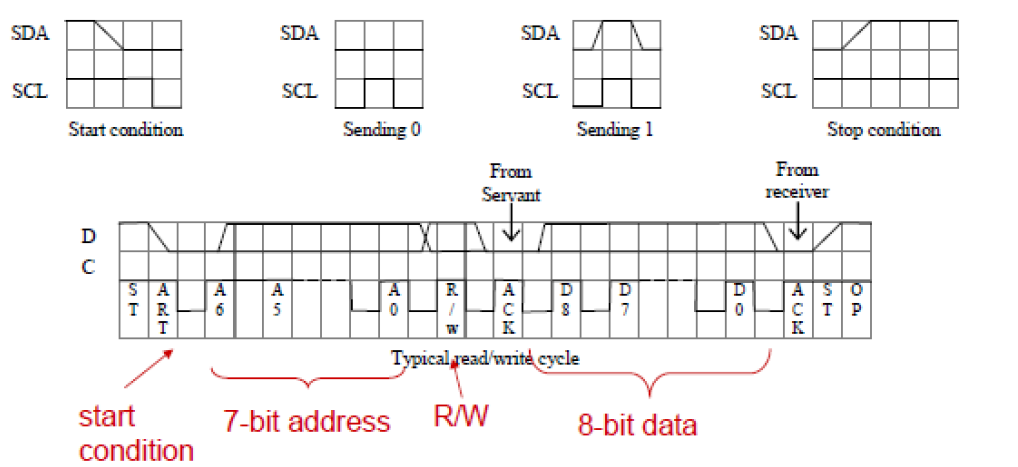
* Подршку за више уређаја;
* Једноставно спајање за различите напоне;
* Понашање као ожичено логичко И;
* Подржава посредовање и решавање судара.

Руководилац контролише (не генерише) такт, и иницира комуникацију, док подређени прима такт и одговара на прозивку руководиоца, али може да обори такт уколико му је потребно више времена. Улоге се могу заменити након стоп секвенце. Сваки уређај повезан на магистралу има јединствену адресу. Формат адресе се састоји од адресног поља:

* Изворно – 7 бита + 1 најнижи бит у адреси за статус уписа или читања (1 означава читање, 0 упис).
* Проширено – 10 бита. Водећи октет се састоји из петобитне контролне секвенце (11110), 2 горња бита адресе и бит за статус уписа или читања (исти начин фунцкионисања као и за претходни тип адресе). Нижи октет представља 8 доњих бита адресе.

Максималан број уређаја на магистрали је ограничен бројем расположивих адреса и укупном капацитивношћу магистрале (максимално 400 pF). Максимална дужина линије је неколико метара. Пошиљаоц поставља податке на *SDA* линију, а прималац их потврђује. Руководилац започиње комуникацију старт секвенцом – поставља *SDA* линију на ниску ивицу док је *SCL* линија на високој ивици. Редовне промене *SDA* се једино дешавају када је *SCL* на ниској ивици. Пренос података се завршава стоп секвенцом. Након старт секвенце шаље се адресно поље, прозвани зависни потврђује; након потврде иду подаци које такође прати потврда. Могуће је слање више података истовремено. Оквир изгледа исто приликом слања и читања, где је статусни бит читања и писања једини различит.

Такође, могућ је и комбиновани пренос који служи за промену смера трансакције, где руководилац уместо стоп секвенце шаље продужени старт, а затим и ново адресно поље са битом за писање или читање. Подешавање бита података се врши за време ниског нивоа такта, а узорковање бита података се врши за време високог нивоа такта. Да би се избегло лажно препознавање пожељно је радити узорковање и подешавање на ивицу такта. Временски дијаграм *I²C* је приказан на слици 2.18 .



Слика . Временски дијаграм *I²C.*

## Упоредно поређење протокола

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *CAN* | *LIN* | *SPI* | *I²C* |
| Брзина | До 1 *Mb/s* | До 20 *kb/s* | До 2 *Mb/s* | До 5 *Mb/s* |
| Број  линија | 2 | 1 | 4 | 2 |
| Предности | Прилагођеност за асинхрони саобраћај.  Висока пропусна моћ под малим оптерећењем.  Ефикасно коришћење пропусног опсега.  Могуће дефинисати локалне и глобалне приорите.  Константан премашај: битска арбитрација део поруке. | Проширење CAN подмреже.  Смањење цене ожичења.  Побољшава ЕМК.  Нема потребе за арбитражом.  Без спољњег кристала.  Смањује ризик од доступности.  Детерминизам и предвидљивост. | Велика брзина за тачка на тачку преносе.  Подршка за симултани улазно/излазни ток података.  Не постоји адресирање; олакшана имплементација.  Широко подржан. | Bише брзине, дуплекс, једноставни померачки регистри. |
| Мане | Потребно доминантно битско стање: немогућност трансформаторског одвајања,  немогућност окидања на време.  Брзина простирања ограничава дужину магистрале.  Неправедан приступ: високи приоритет може да окупира мрежу.  Повећана кашњења за низак приоритет: изгладњивање  Потребно филтрирање да се не би преливали спремнике. | Уређаји су подобни раличитим врстама малициозних напада. | Руковање више подређених: усложњава линије.  Немогућност комуникације између подређених.  Нема потврде пријема: такт могуће да одлута  Не постоји решавање колизија.  Не постоји контрола тока (потребно знати брзину подређених). | Ограничења за проширења: број уређаја ограничен новим линијама.  Тешко имплементирати само у софтверу.  Већи број линија. |

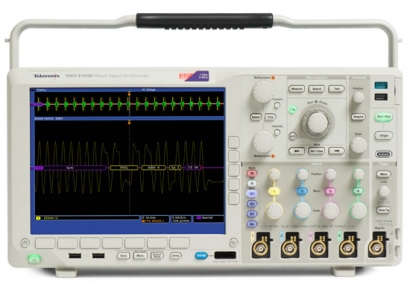
Као резиме претходних поглавља у табели 2 је приказан упоредни приказ протокола.

Табела 2 Упоредно поређење протокола.

# Концепт решења

## Осцилоскоп

Приликом израде овог дипломског рада коришћен је *Tektronix DPO 4104B* осцилоскоп, приказан на слици 3.1. Овај осцилоскоп има четири независна канала. Сондама су мерени напонски нивои који су одређени сигнали протокола имали, чије су жице претходно повезане на прототипску плочу. Коришћен је један или два канала у зависности од протокола који се користи. Окидачи су потом подешени за нивое измереног напона, хоризонталну и вертикалну скалу. На самом почетку ово подешавање је било мануелно, праћењем упутства за употребу овог осцилоскопа [9], потом аутоматски – комуникацијом преко *USB*-а употребом *API*-ја.



Слика 3.1 *Tektronix DPO 4104B.*

## Повезивање *CAN* – и мерење са осцилоскопом

За генерисање сигнала за *CAN* магистралу коришћен је *Vector VN8912A* модул и алат *CANoe*. *CANoe* алат је напредни алат за развој, тестирање и анализу како појединачних електронских управљачких јединица, тако и комплетне мреже. Овим алатом могуће је заокружити комплетан процес од планирања мреже до тестова на системском нивоу. Корисници *CANoe*-а могу анализирати више магистрала као и цео систем на свом столу.

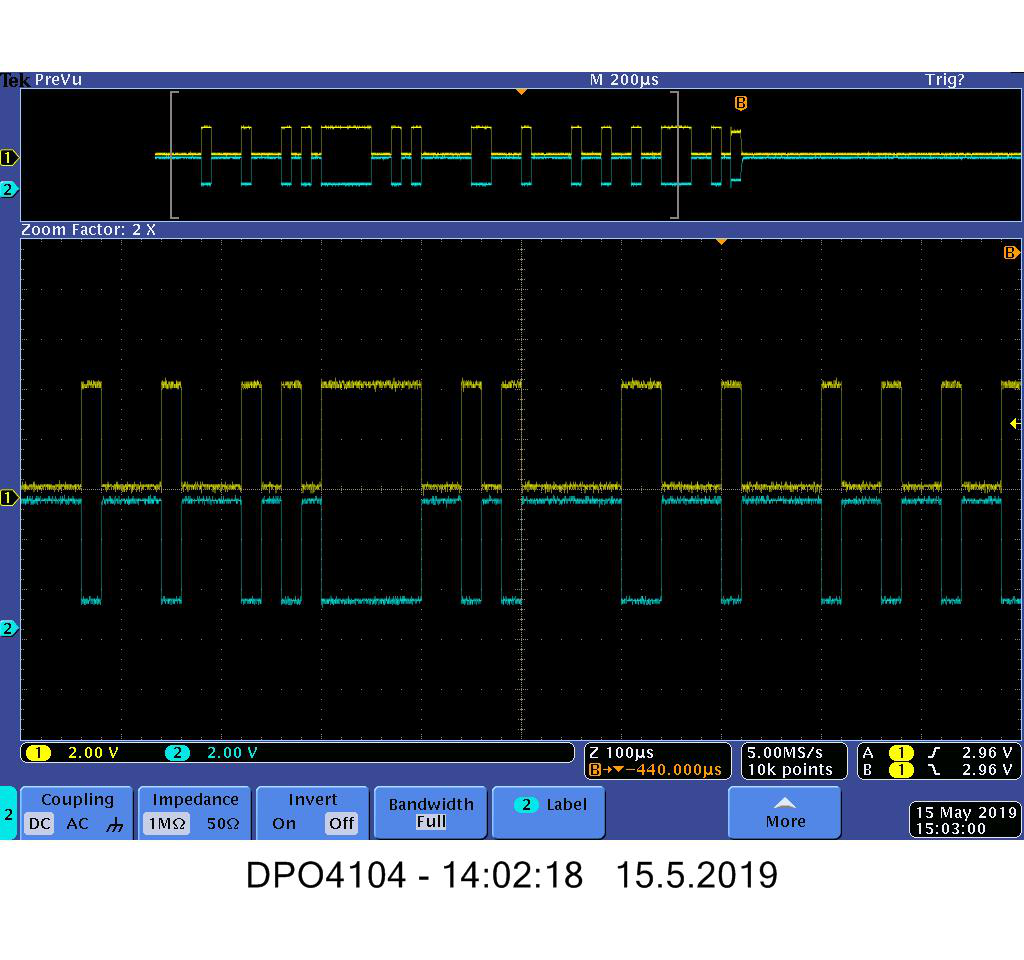
Основне предности овог алата су:

* Један заједнички алат и за тестирање и за развој;
* Лако аутоматско тестирање;
* Могућност симулирања и тестирања електронских управљачких јединица преко дијагностике;
* Откривање и исправљање грешака у раним фазама развоја софтвера;
* Интуитивно графичко окружење и евалуација резултата базирана на тексту.

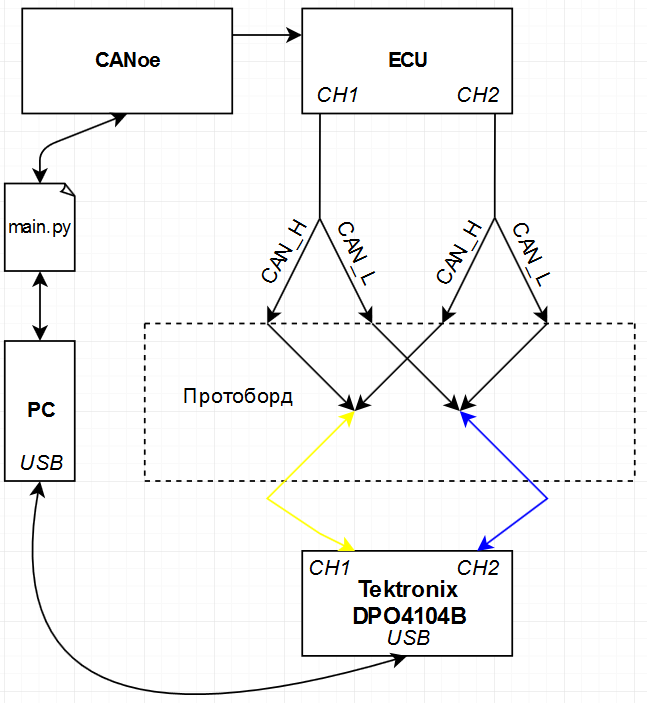
У *CANdb++ Editor*-у је креирана база података (формат .*dbc*) која описује целокупну CAN мрежу. Свака *CAN* мрежа је дефинисана са:

* мрежом;
* контролоном јединицом;
* променљивама окружења;
* мрежним чворовима;
* порукама;
* сигналима.

Само креирање ове базе података ће бити детаљније објашњено у програмском решењу. Након подешавања конфигурације у *CANoe*-у и укључивања креиране базе података, урађено је повезивање модула *VN8912A* и рачунара *USB* каблом, а затим је покренута симулација док су са друге стране повезане две пробе осцилоскопа на   
*CAN-H* и *CAN-L* канал чије су жице изведене и прикључене на прототипску плочу. Окидач за напон је подешен на 2.96 V, скала на 100 микросекунди, док је сваки од два канала скалиран на 2 V. Екран осцилоскопа приликом снимљеног *CAN* сигнала је приказан на . Дијаграм који објашњава повезивање *CAN*-а је приказан на .



Слика . Приказ екрана осцилоскопа приликом снимања *CAN* сигнала

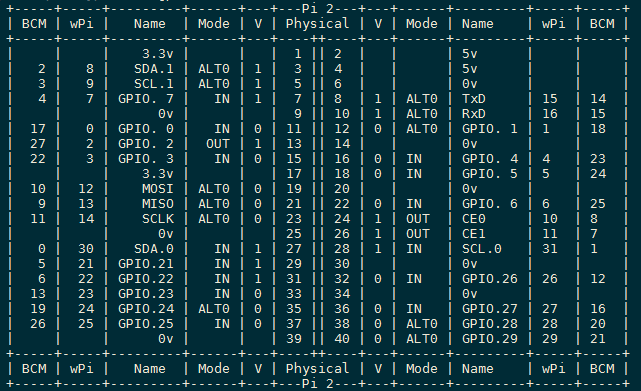


Слика . Дијаграм повезивања *CAN.*

## Повезивање *LIN* – а и мерење са осцилоскопом

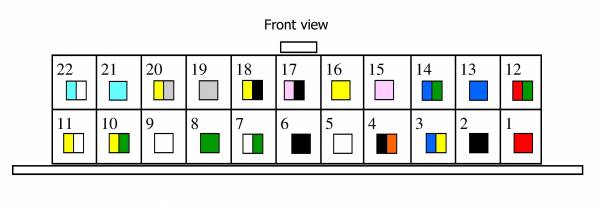
*Carberry* је аутомобилска плочица за проширење рачунара *Raspberry Pi*. Идеја је да се понуди спона ка електроници и комуникационим каналима у аутомобилима и самим тим омогући интеракција, али и развој апликација за крајњег корисника као што су забавно-информациони системи (медија центар, дијагностика аутомобила, логовање података, управљање флотом возила, праћење возила), алармни системи, пружање Интернета и слично. Такође *Carberry* може да послужи и у образовне сврхе да би се проучавали аутомобилске магистрале и протоколи.

На самом почетку је *Carberry* плочица повезана преко 40-пинског *GPIO* интерфејса на *Raspberry Pi 2* модел Б рачунар. Преглед пинова *Raspberry Pi 2* модел Б рачунарa (добијено куцањем команде *gpio readall*) је приказан на слици 3.4. *Carberry* комуницира са *Raspberry Pi* употребом пинова 8 (за слање података), 10 (за пријем података) ,12, 13 (за гашење) на *Raspberry Pi* –у*.*



Слика . *Raspberry Pi 2* модел Б, преглед пинова.

Следећи корак је био повезивање жице са главне утичнице *Carberry* модула на прототипску плочу. Илустровани приказ главне утичнице се налази на слици 3.5, а табела са описом свих сигнала се налази у табели 3 где су црвеном бојом означени каблови за напајање и уземљење, а зеленом каблови за *LIN* (остали каблови ми нису релевантни тако да њихова улога није описана). Након што су повезана два оваква модула на прототипску плочу, комуникација је остварена преко серијског порта коришћењем *microUSBТоUSB* кабла. Након тога је започета комуникација са сваким од ових модула. То је урађено коришћењем команди које су стандардизоване од стране произвођача *Carberry*-ја.

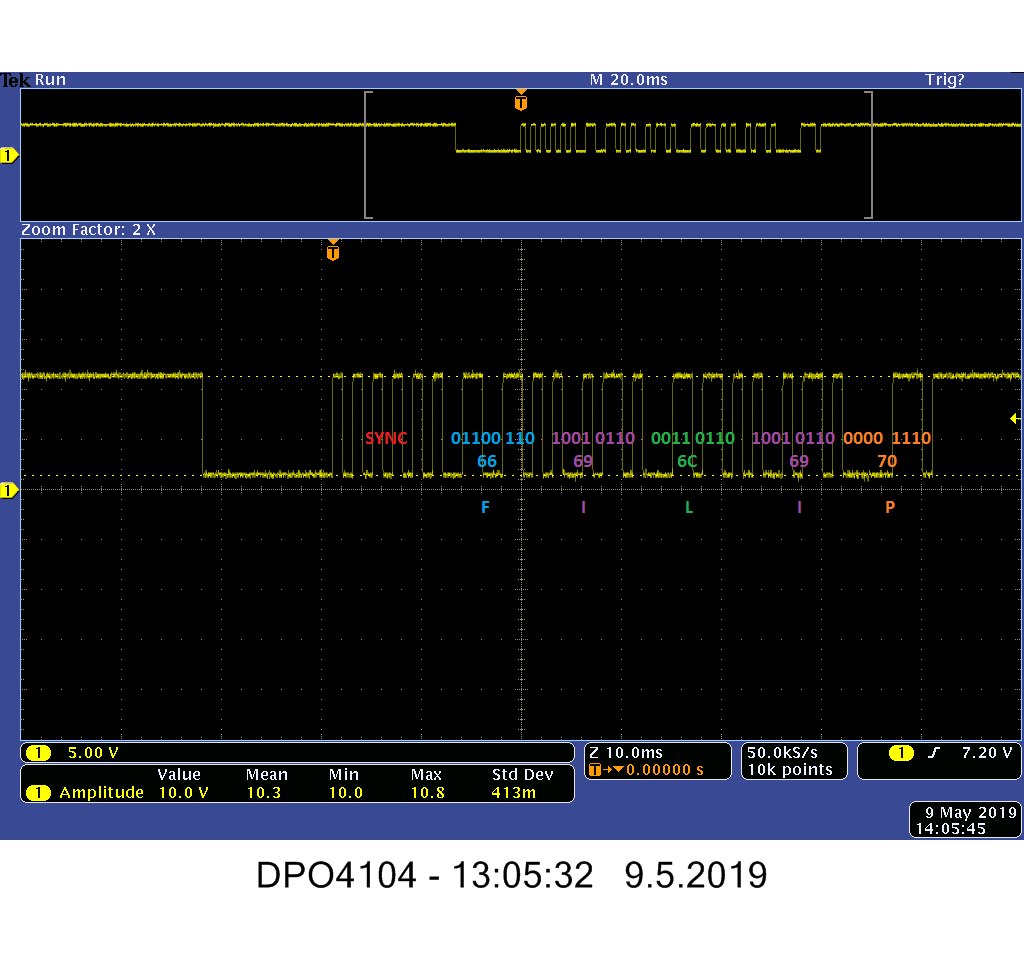


Слика . Илустровани приказ главне утичнице за *Carberry* модул.

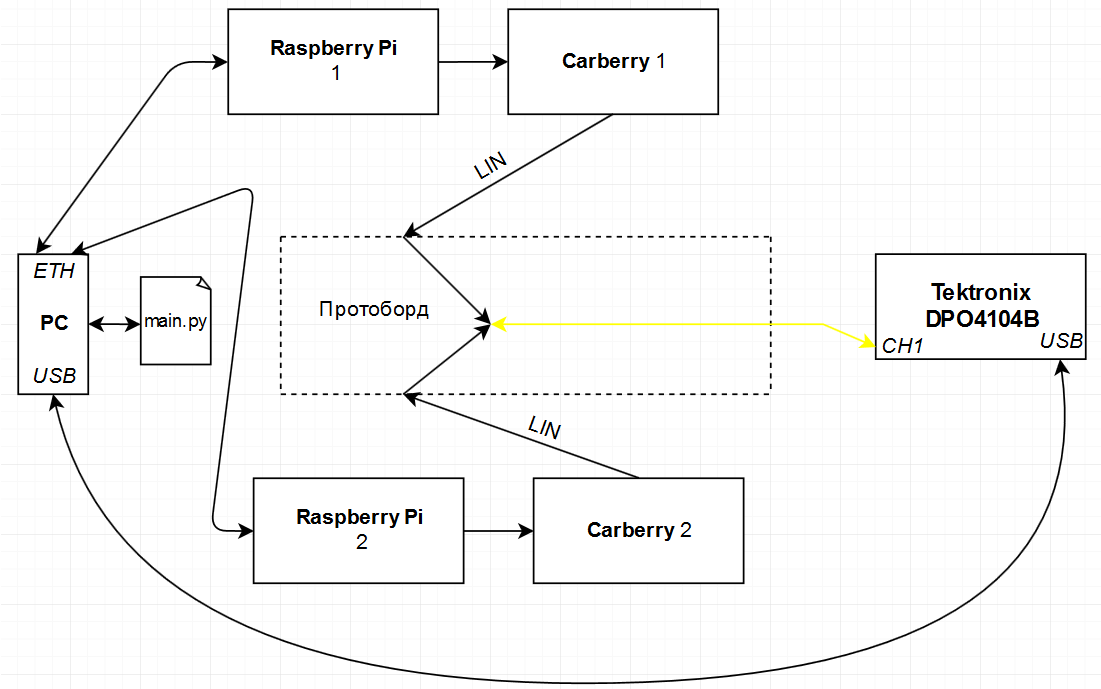
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Позиција** | **Боја жице** | **Опис** | **Позиција** | **Боја жице** | **Опис** |
| **1** | **Црвена** | **+12V Напон** | **12** | **Црвено**  **/Зелена** | **LIN** |
| **2** | **Црна** | **Узмемљење** | 13 | Плава | / |
| **3** | **Плаво**  **/Жута** | **+12V Улаз за паљење** | 14 | Плаво  /Зелена | / |
| 4 | Црно  /Наран. | / | 15 | Ружичаста | / |
| 5 | Бела | / | 16 | Жута | / |
| 6 | Црна | / | 17 | Ружичасто  /Бела | / |
| 7 | Бело  /Зелена | / | 18 | Жуто  /Црна | / |
| 8 | Зелена | / | 19 | Сива | / |
| 9 | Бела | / | 20 | Жуто  /Сива | / |
| 10 | Жуто  /Зелена | / | 21 | Светло плава | / |
| 11 | Жуто  /Бела | / | 22 | Светло-плава/  Бела | / |

Табела 3 Опис жица главне утичнице за *Carberry* модул.

Ове команде ће детаљније бити објашњене у програмском решењу. Оба уређаја се морају подесити на истој брзини комуникације. Након тога се могу слати команде за генерисање сигнала. На осцилоскопу је скала подешена на 10 ms, окидач на 7.2 V а скала самог сигнала на 5 V. Пример сигнала који је снимљен осцилоскопом је приказан на слици 3.6. Дијаграм повезивања за *LIN* је приказан на слици 3.7.



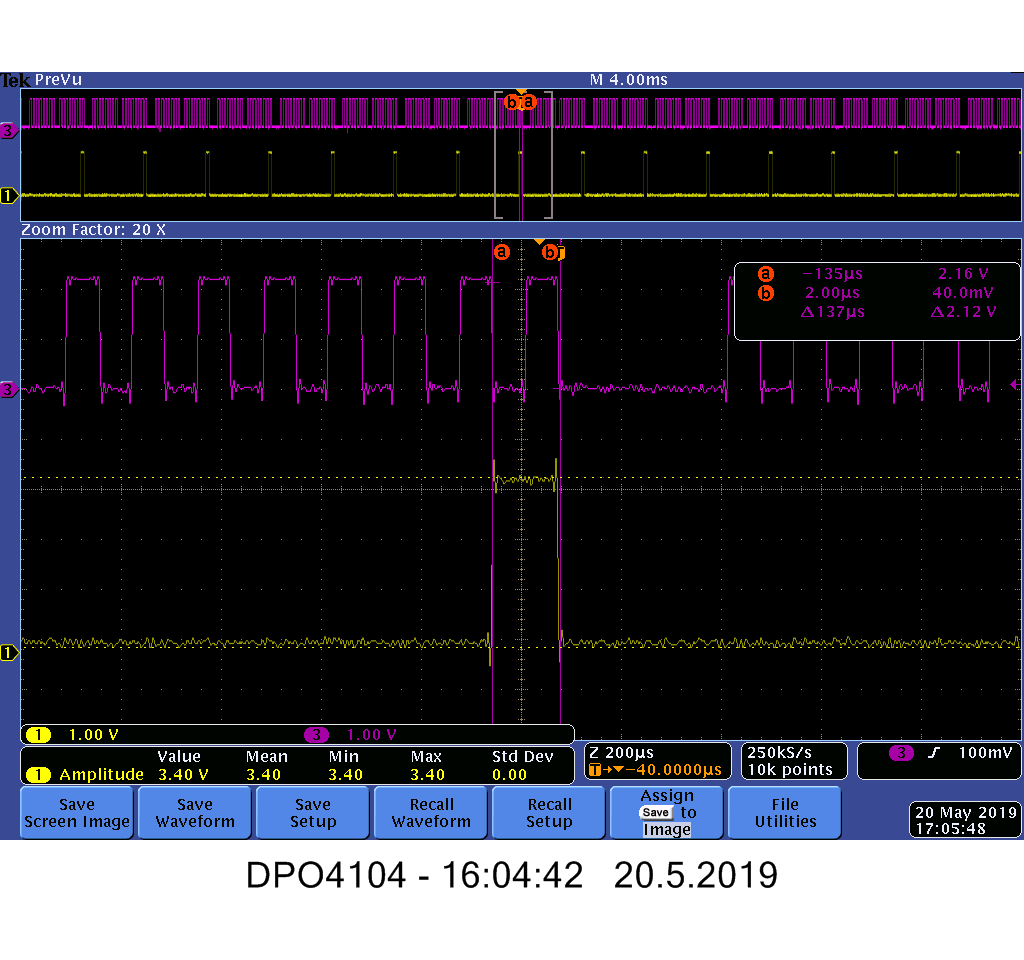
Слика . Приказ екрана осцилоскопа приликом снимања *LIN* сигнала.



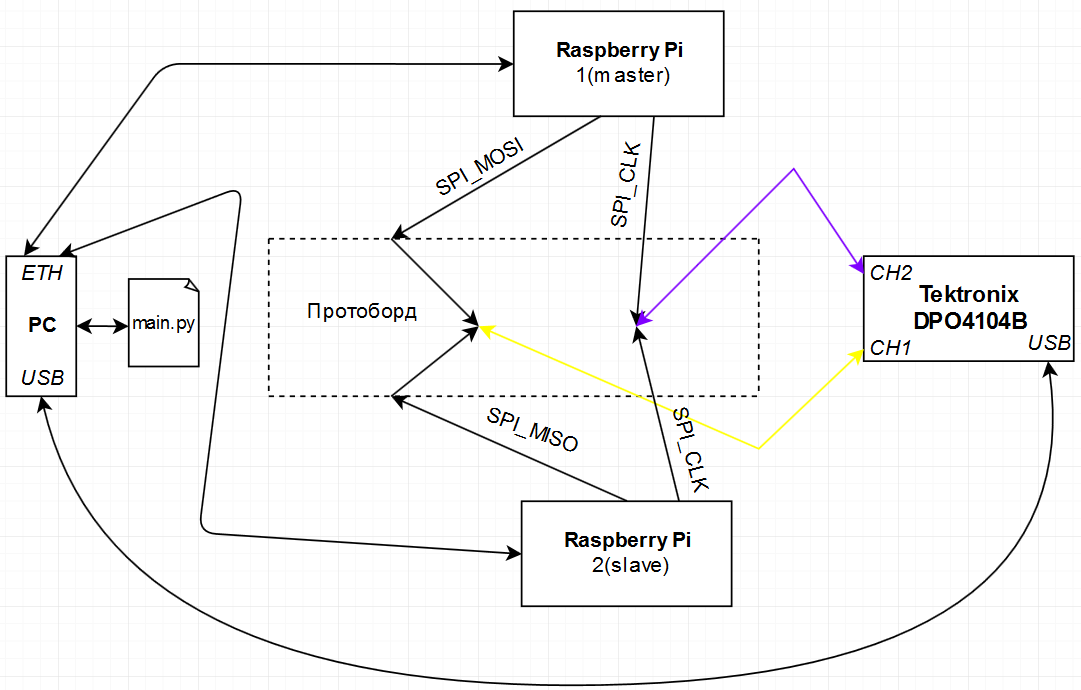
Слика . Дијаграм повезивања *LIN*-a.

## Повезивање *SPI* – и мерење са осцилоскопом

Повезивање *SPI* на прототипску плочу се врши праћењем пинова за *GPIO* интерфејс *Raspberry* *Pi* рачунара као на слици 3.4, при чему су коришћени само пинови *GPIO* 9 (*SPI\_MOSI*), *GPIO* 10 (*SPI\_MISO*) и 11 (*SPI\_CLK*). Један *Raspberry Pi* треба да се понаша као надређени, док је други подређени. Међусобно на прототипску плочу је повезана *SPI\_MOSI* линија надређеног уређаја са *SPI\_MISO* линијом подређеног уређаја и *SPI\_CLK* линија једног уређаја са *SPI\_CLK* линијом другог уређаја. Потом се употребом *SSH* протокола врши комуниција са *Raspberry Pi* (надређеним уређајем) и покретањем *Python* скрипте (која ће детаљније бити објашњена у програмском решењу) се генерише секвенца *SPI* сигнала. Пример снимљеног *SPI* сигнала је приказан на слици 3.8. Окидач је подешен на 100 mV док се скала подешава на 200 микросекунди. Скала сваког од канала се подешава на 1 V. Дијаграм повезивања *SPI* је приказан на слици 3.9.



Слика . Приказ екрана осцилоскопа приликом снимања *SPI* сигнала.



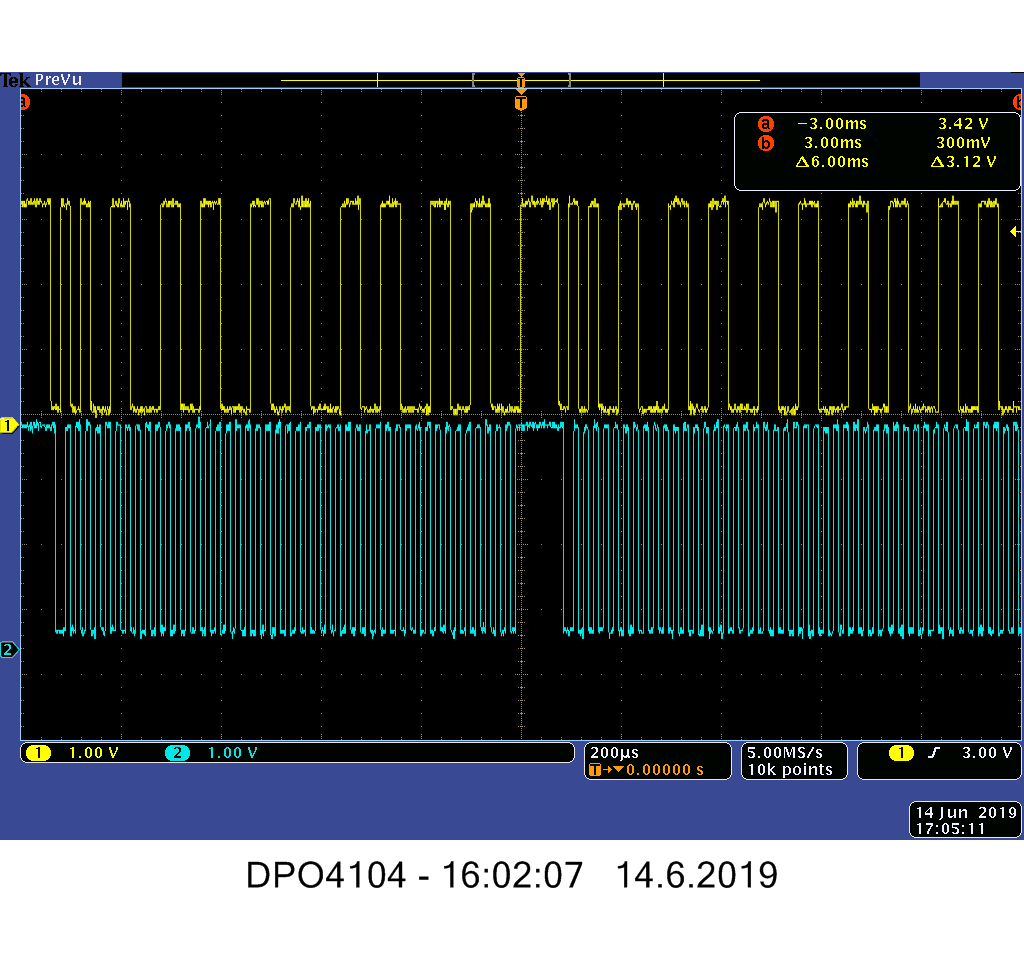
Слика . Дијаграм повезивања *SPI.*

## Повезивање *I2C –* и мерење са осцилоскопом

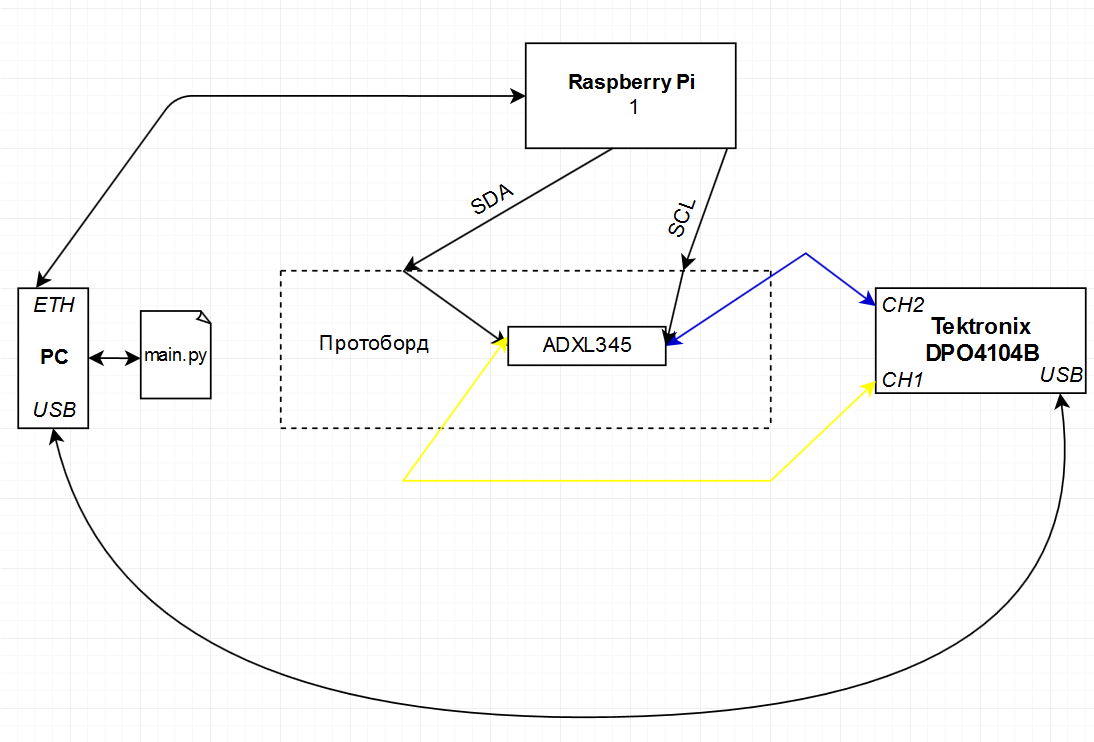
Повезивање *I2C* се врши коришћењем пинова 1 (*DC Power*), 3 (*SDA*), 5 (*SCL*) и 30 (*Ground*). Због тога што овај протокол на *Raspberry Pi* захтева уређај повезан на ове пинове за успешну детекцију саобраћаја на мрежи, користи се *ADXL345* сензор за акцелерометар произведен од стране *Sunfouder* компаније.

Потом, коришћењем две сонде вршено је очитавање сигнала, при чему је намена сонде на каналу 1 осцилоскопа била да прикупи сигнал са подацима (*SDA*), а сонде на другом да прикупи сигнал са радним тактом (*SCL*). Пример снимања ова два сигнала осцилоскопом је приказан на слици 3.10.

Окидач за сваки од канала је подешен на 3 V док је скала подешена на 200 микросекунди. Дијаграм повезивања *I2C* је приказан на слици 3.11.



Слика . Приказ екрана осцилоскопа приликом снимања *I2C* сигнала.



Слика . Дијаграм повезивања *I2C.*

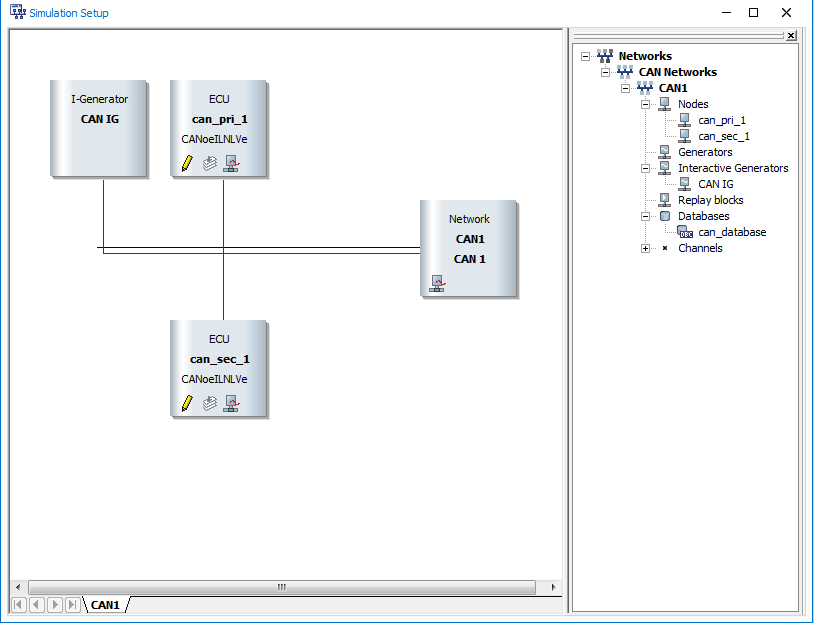
# Програмско решење

## Комуникација са осцилоскопом, мерење и обрада сигнала

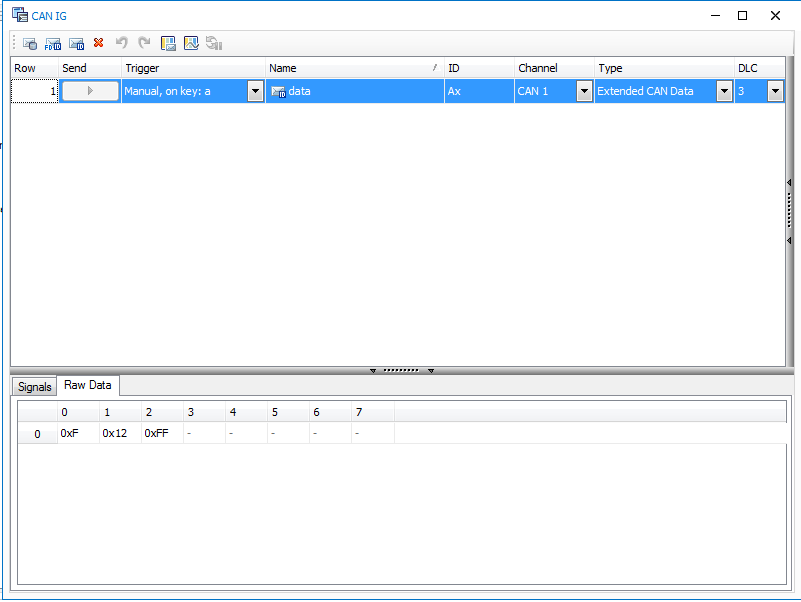
Комуникација са осцилоскопом је урађена коришћењем *Python PyVisa* модула. Овај модул омогућава иницијализацију осцилоскопа као објеката класе *ResourceManager*. Функцији *open\_resource* овог објекта као параметар се шаље идентификациони број осцилоскопа који је приказан у апликацији *OpenChoiceDesktop*. Размењивање података са осцилоскопом се своди на две методе објекта – *write()* и *ask().* Праћењем програмерског приручника *Tektronix*-а [10] за овај модел осцилоскопа послате су команде као параметар једној од две претходно описане фунције, у зависности од намене. Послате су команде за одабир канала, постављање хоризонталне и вертикалне скале, а потом за прикупљање података. *CANoe*

У *CANoe* је прво креирана одговарајућа конфигурација. Потом је у *CANoe* Options менију под *Measurment* опцијом, и у *General* прозору *Channel Usage,* подешен на одговарајући број *CAN* канала (2). У *Simulation Setup* прозору су потом креирана два *ECU* чвора и један *I-Generator* који су повезани на претходно направљену *CAN* мрежу. *Simulation Setup* је приказан на . *I-Generator* прозор је приказан на .  
У менију *Hardware* под опцијом *Network Hardware* је намештена одговарајућа бодна брзина која мора да одговара бодној брзини која се користи за декодовање. Ако ово није правилно подешено, декодовање неће бити исправно. Потом је креирана база података за *CAN* мрежу у *CANoe Candb++*, унутар које су креирана 2 чвора (примарни и секундарни).

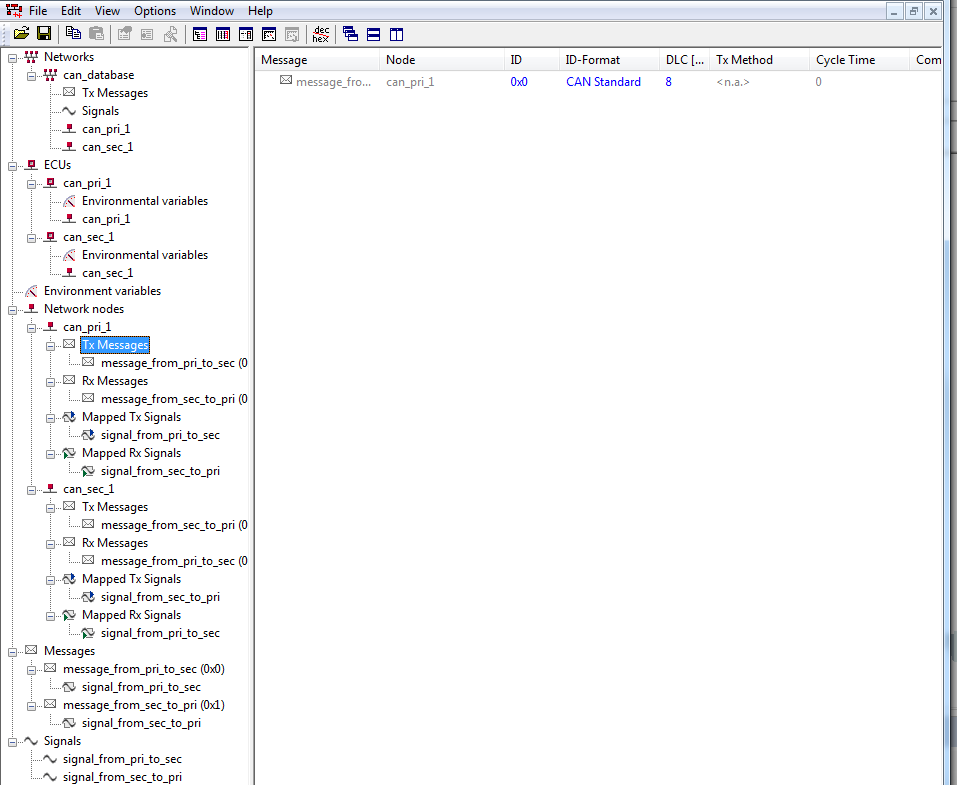
Након тога су креирана два сигнала, при чему је један резервисан за слање података од примарног чвора до секундарног, а други за слање обрнутим смером. Затим су ови сигнали мапирани за одговарајућу поруку која је претходно креирана. На крају су ове поруке мапиране на сваки од одговарајућих чворова, чиме се завршава креирање базе података. *Canoe CANdb++* *Editor* је приказан на слици 4.3 . Унутар *I-Generator***-**а су креирани сигнали који ће се слати од примарног до секундарног чвора. За прављење сигнала мора се одабрати канал, *ID*, тип сигнала (да ли је са стандардним идентификатором или продуженим), дужина података, а унутар *Raw Data* се конфигурише сваки од октета за слање. Пре самог покретања (*.cfg)* конфигурације, мора се потврдити да унутар *Home* прозора буде подешено да је конфигурација у *Online Mode* и да се користи *Real Bus*, пошто ће се сигнал генерисати хардверски, коришћењем модула *VN8912A*. Праћење сигнала у времену се врши унутар *Data* прозора из *Analysis* менија.



Слика . Приказ *Simulation* *Setup* прозора унутар *CANoe*-а.



Слика . Приказ *I-Generator* прозора.



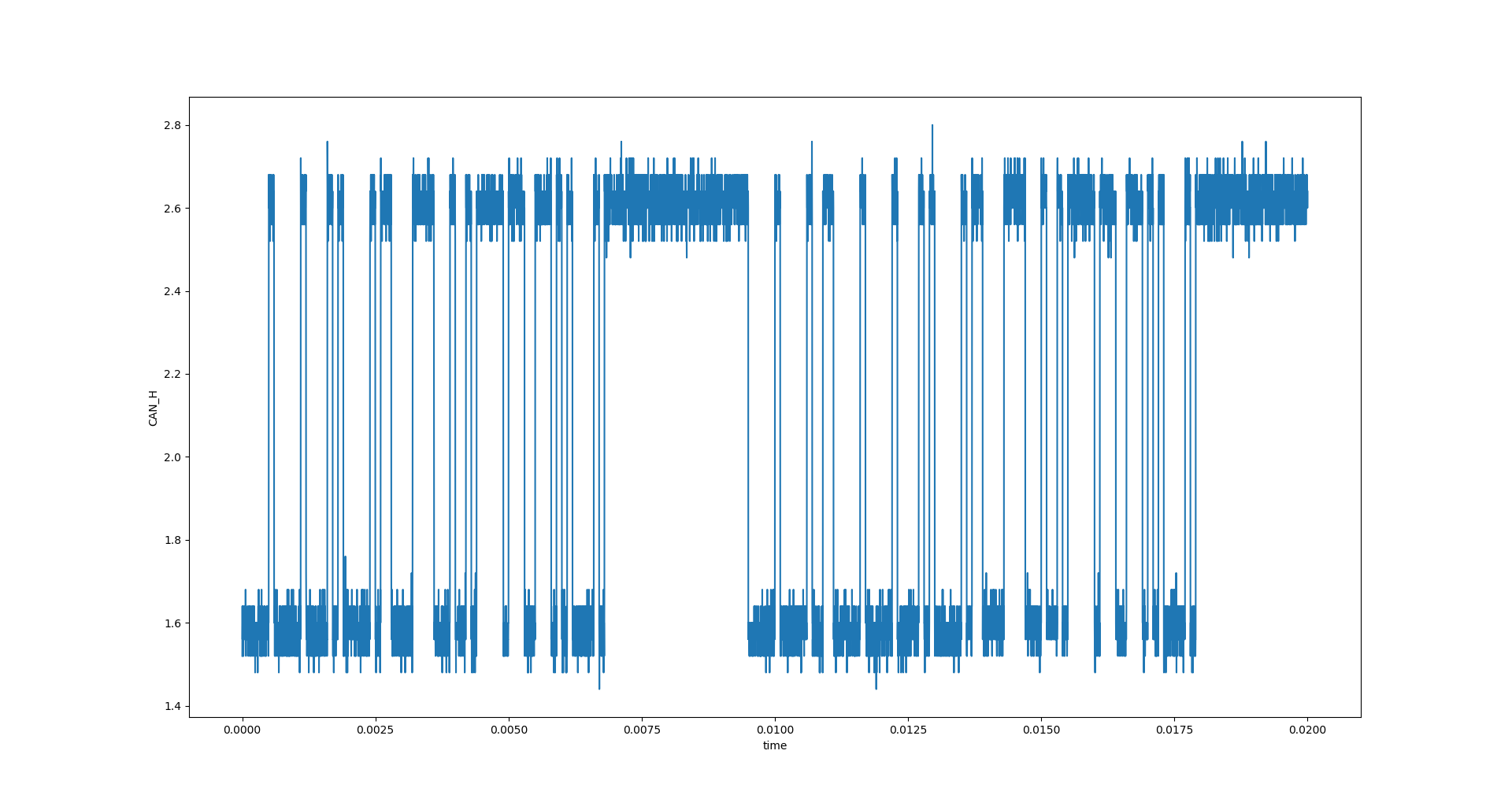
Слика . Приказ *Canoe CANdb++ Editor*-а.

## Декодовање *CAN*-а

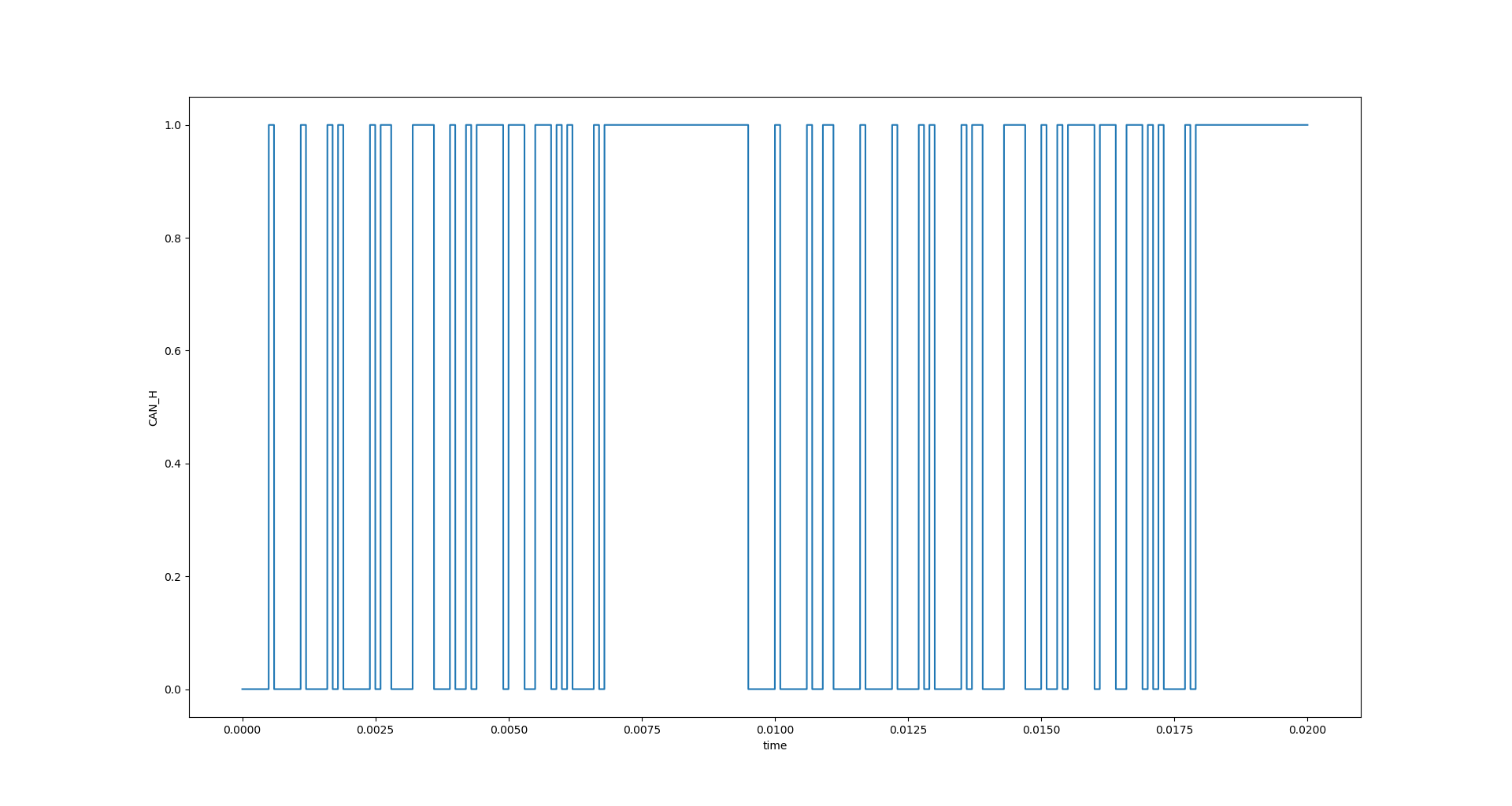
Декодовање *CAN* сигнала се врши поштујући *CAN 2.0* протокол. Напони прикупљеног сигнала се представљају у облику 1 и 0, при чему су сви делови сигнала са напоном изнад претходног постављеног окидача постављени на 1, док су остали постављени на 0. Након тога су избачени из сигнала сви убачени и лавински бити. Коришћењем *Knuth-Morris-Pratt* алгоритма се препознавају лавински бити и 5 истих доминантних или рецесивних бита, и за свако место у низу где су се појављивали ови бити долази до брисања података – тек након овога се могло наставити са декодовањем при чему се гледа бит 13 сигнала који указује на то да ли је коришћен *CAN 2.0 A* или *CAN 2.0 B*.

Потом се издвајају сви релевантни оквири и израчунава се циклична провера редудансе. Ако се ова израчуната провера поклапа са цикличном провером редудансе издвојеном из сигнала, главном програму се прослећује декодован *CAN* сигнал са назнаком да је провера тачна (оквири су означени зеленом бојом), у супротном, прослеђује се декодован *CAN* сигнал али означен црвеном бојом.

Ухваћен *CAN* сигнал, необрађен и обрађен, приказани су на слици 4.4 и слици 4.5 , респективно.



Слика . Приказ ухваћеног необрађеног CAN сигнала са осцилоскопа.



Слика . Приказ ухваћеног *CAN* сигнала са осцилоскопа након обраде.

## Декодовање *LIN* – а

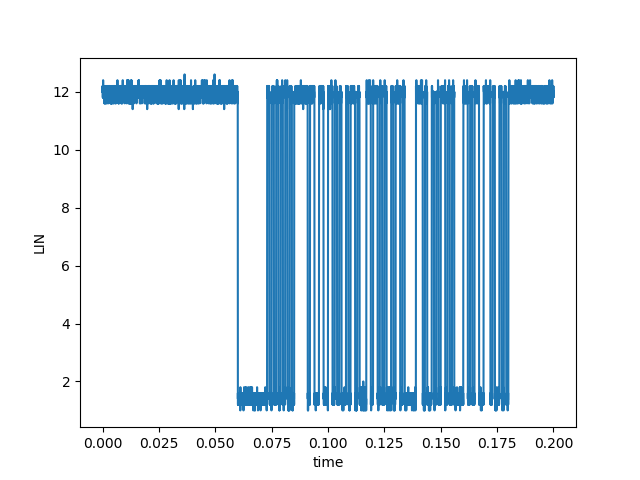
Као што је наведено у концепту решења, у овом поглављу биће дато објашњене стандардне команде за *Carberry* за *LIN* протокол. Прво, преко серијског порта, потребно је остварити конекцију за сваки од *Carberry* модула коришћењем *LIN OPEN* команде. Постоји пет различитих режима рада које су подржане од стране *Carberry*-ја (табела 4).

| **Mод** | **Команда** | **Опис команде** |
| --- | --- | --- |
| *Master 1x* | *LIN OPEN MASTER1X*  <бодна брзина> | Отвара канал при чему аутоматски препознаје брзину на магистрали, уређај се понаша као надређени, користи се стандардна контролна сума. |
| *Master 2x* | *LIN OPEN MASTER2X,*  <бодна брзина> | Отвара канал при чему аутоматски препознаје брзину, уређај се понаша као надређени, користи се продужена контролна сума. |
| *Slave* *1x* | *LIN OPEN SLAVE1X* | Отвара канал при чему аутоматски препознаје брзину, уређај се понаша као подрећени, користи се стандардна контролна сума. |
| *Slave* *2x* | *LIN OPEN SLAVE2X* | Отвара канал при чему аутоматски препозанаје брзину, уређај се понаша као подрећени, користи се продужена контролна сума. |
| *Free* | *LIN OPEN FREE* <бодна брзина> | Отвара канал и задаје брзину; за паритет и конролну суму је одговоран корисник. |

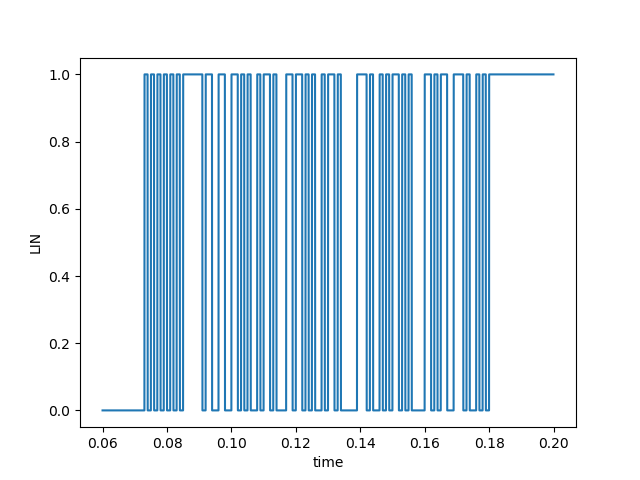
Табела 4 *Carberry* команде за *LIN*

У изради рада коришћен је последњи, „слободни“ режим због флексибилности. Након слања сваке команде, ако је она исправна, добија се одговор *OK*. Слање *LIN* оквира у слободном режиму се ради употребљавањем *LIN TX* <заштићени идентификатор> [*D0…D7*] [контролна сума] команде. Да би се проверило да ли је контролна сума исправна, израчуната је независно праћењем упутства из *LIN* спецификације. Потом се пореди да ли израчуната контролна сума одговара контролној суми у оквиру. Такође се мора проверити паритет за сваки од послатих оквира. У зависности од тога да ли су подаци послати у оквиру одговарали израчунатим подацима, на стандардном излазу екрана се шаље обавештење да ли има грешке у послатом оквиру.

Пример ухваћеног *LIN* сигнала, необрађеног и обрађеног, приказан је на сликама 4.6 и 4.7, респективно.



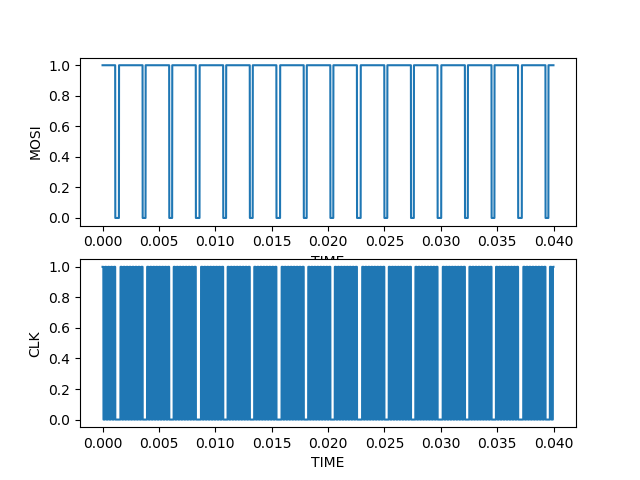
Слика . Приказ необрађеног *LIN* сигнала.



Слика . Приказ обрађеног *LIN* сигнала.

## Декодовање *SPI*-а

Да би се препознали подаци који треба да се декодују, прво су се морали издвојити корисни подаци на *MOSI* линији који се шаљу у исто време када се појави секвенца од 8 растућих ивица на *SCLK* линији. За претрагу секвенце на *SCLK* линији се користи *Knuth-Morris-Pratt* алгоритам. Уколико секвенца није правилно пронађена  
(ако није правилно подешена скала) добиће се корумпирани подаци. Уколико је секвенца пронађена портребно је посматрати податке који имају дужину ове секвенце. Корисне податке можемо да издвојимо тек након што је утврђено да је дужина секвенце правилна. Након што су издвојени корисни подаци, исти се декодују и представљају у хексадецималном формату. Пример обрађеног *SPI* се може видети на слици 4.8.

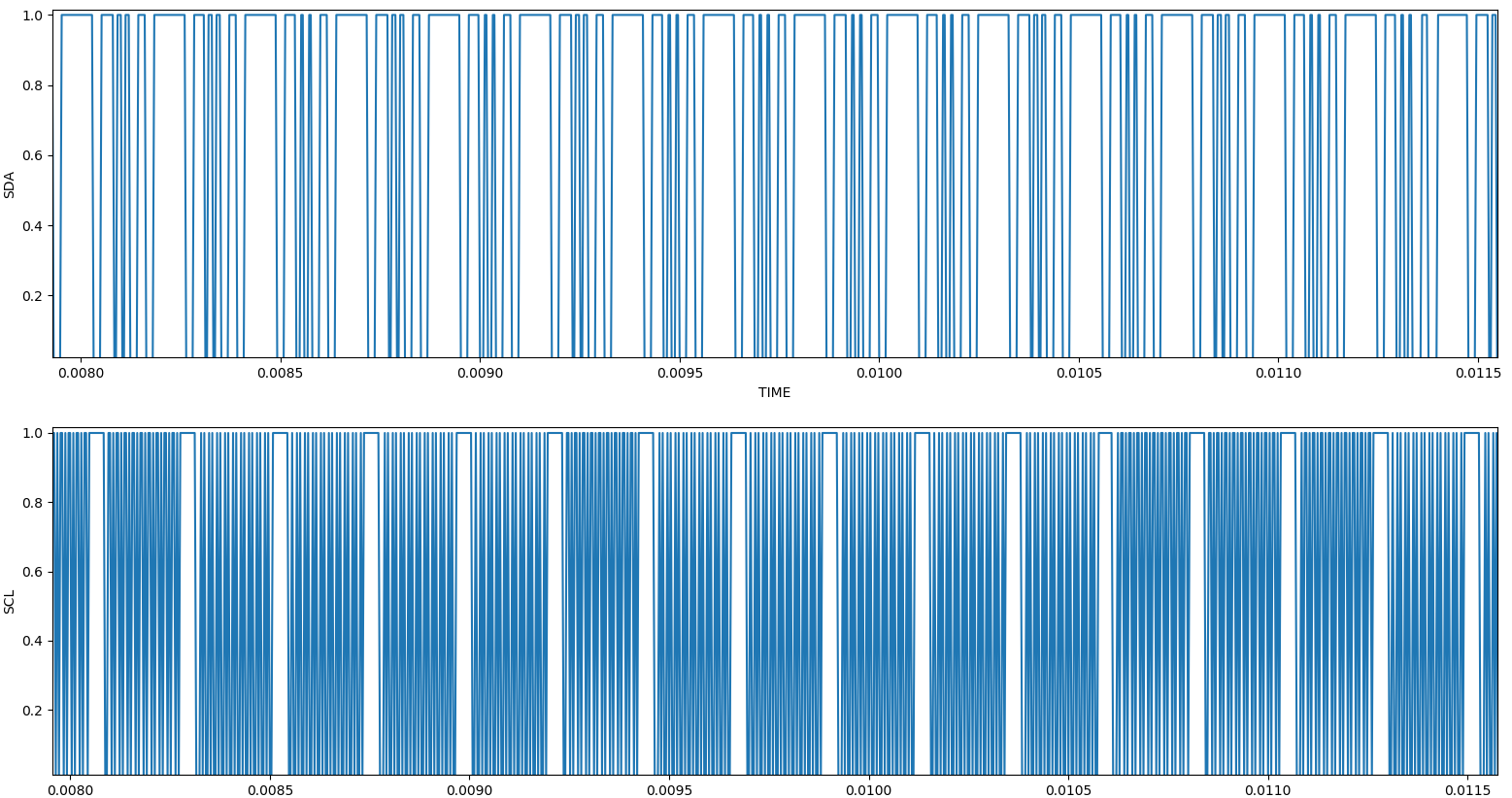


Слика . Пример обрађеног *SPI* сигнала

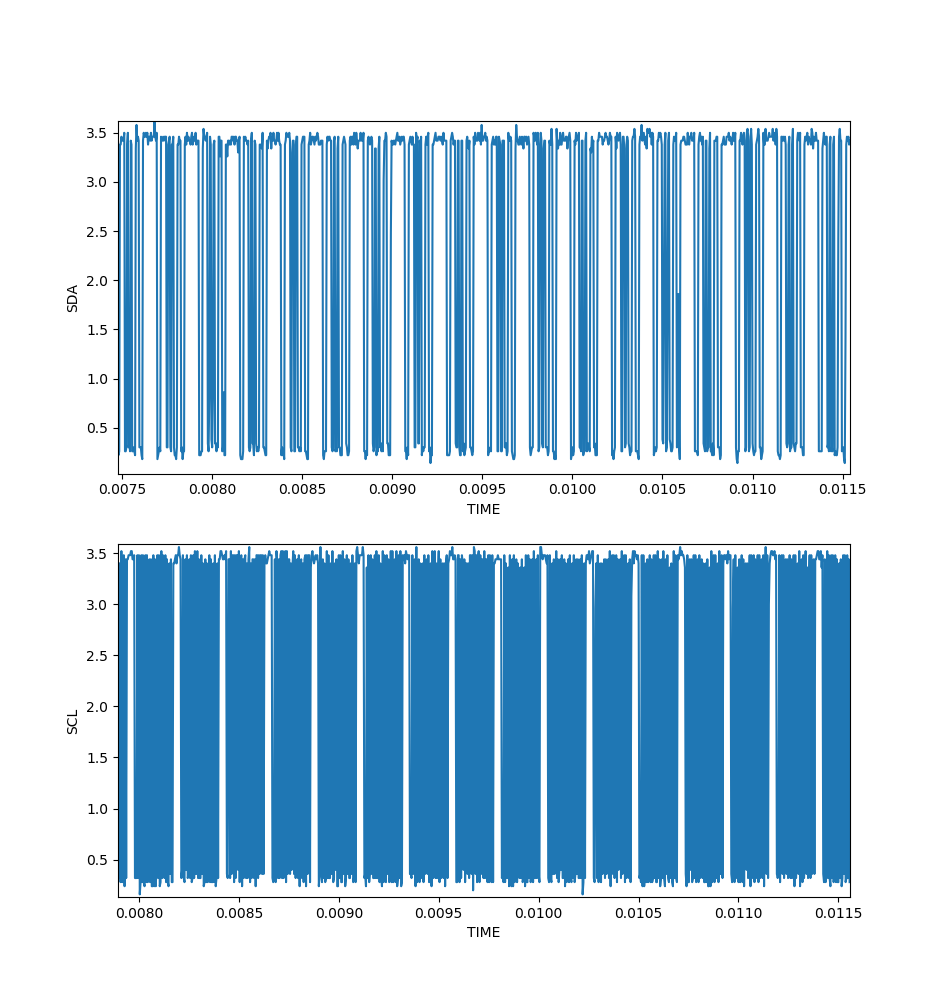
## Декодовање *I2C*-а

Декодовање *I2C* сигнала је започето претрагом почетне секвенце за сваки од канала (*SDA*, *SCL*) – почетна секвеца канала за пренос радног такта је за један бит већа. За претрагу ове две секвенце је коришћен *Knuth-Morris-Pratt* алгоритам.

Приказ необрађеног сигнала је на . Приказ ухваћеног сигнала који је обрађен и спреман за декодовање је приказан на .



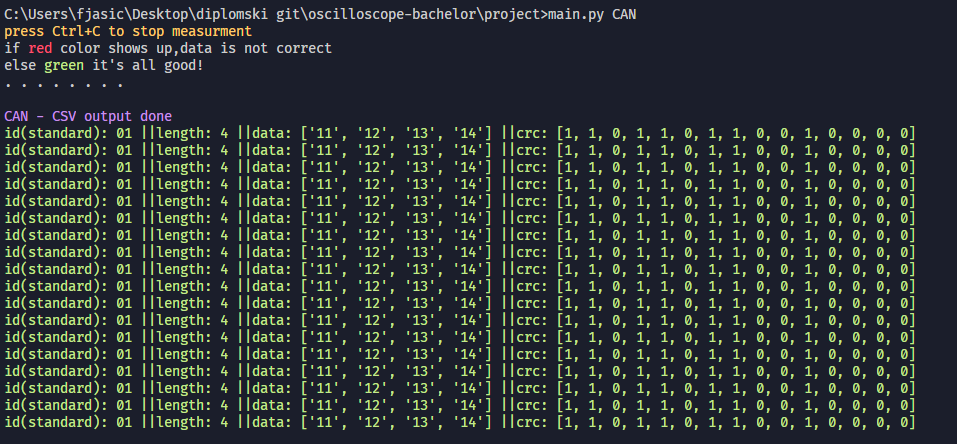
Слика . Приказ обрађеног *I2C* сигнала.



Слика . Пример необрађеног *I2C* сигнала.

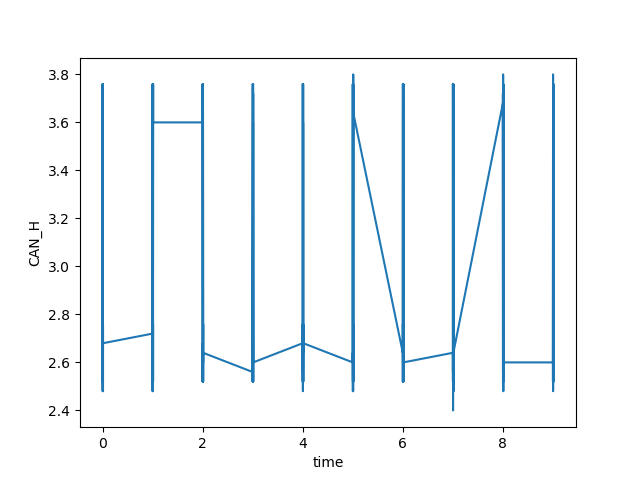
# Тестирање и резултати

Провера исправности имплементације декодера за протоколе је урађена покретањем генератора *CAN, LIN, SPI и I2C* оквира, и потом, посматрањем излаза покренуте апликације за декодовање. Приликом покретања апликације за декодовање, потребно је као аргумент унети тип протокола (у случају *LIN*-a, потребно је унети и коју врсту контролне суме желимо да декодујемо *c –* за класичну контролну суму*,e* – за проширену контролну суму), као и режим у којем се извршава (*online –* када се жели очитавање директно са осцилоскопа или *offline -* када се жели очитавање из једне од *.csv* датотека - име је креирано у зависности од тога који се протокол користи, чији се садржај ствара при сваком покретању апликације у *online* режиму). Генерисање *CAN* оквира је урађено у *CANoe* - у на начин описан у претходном поглављу. Генерисање *LIN* оквира је урађено у *serial\_lin.py*. Генерисање *SPI* оквира је урађено у *sendSPI.py*, док је генерисање *I2C* оквира урађено у *sendI2C.py*. Пример излаза успешног декодовања је дат на слици 5.1.



Слика . Пример излаза успешног декодовања.

Потешкоће приликом израде су настале при самом мерењу и обради података, јер при континуалном прикупљању података долази до кашњења између два оквира. Ово је проузроковано коришћењем *Python***-**а који, иако олакшава имплементацију програма, уводи проблеме кашњења и нетачности приликом мерења. Пример кашњења између два *CAN* оквира може се видети на слици 5.2. Кашњење између оквира је око једне секунде.



Слика . Кашњење између *CAN* оквира.

# Закључак

У овом раду је реализовано декодовање *CAN, LIN, SPI* и *I2C* оквира. Функционалност имплементираног декодера олакшава анализирање сигнала који се често користе у аутомобилској индустрији, тиме што корисник овог програма не мора да константно води рачуна о подешавању самог осцилоскопа, већ је подешавање и анализа оквира аутоматски урађена.

Даљи развој ће се фокусирати на реализацију аутоматске детекције тренутног проткола (комплетна аутоматизација програмског решења) као и на додатне функционалности попут: детекција максималне и минималне амплитуде сигнала, и фазе сигнала.

# Литература

[1] Automotive industry. [Online].

Доступно: [https://en.wikipedia.org/wiki/Automotive\_industry](https://en.wikipedia.org/wiki/Automotive_industry%20) [приступљено: јул 2019.]

[2] CAN bus. [Online].

Доступно: <https://en.wikipedia.org/wiki/CAN_bus> [приступљено: јул 2019.]

[3] “BOSCH CAN Specification Version 2.0”, 1991. [Online].

Доступно: [https://www.academia.edu/9998014/BOSCH\_CAN\_Specification](https://www.academia.edu/9998014/BOSCH_CAN_Specification%20) [приступљено: јул 2019.]

[4] ISO 11898-1 “Road vehicles — Controller area network (CAN) — Part 1:

Data link layer and physical signalling”, First edition, Geneva, Switzerland, ISO, 2003.

[5] Local Interconnect Network. [Online]

Доступно: <https://en.wikipedia.org/wiki/Local_Interconnect_Network> [приступљено: јул 2019.]

[6] “LIN Specification Package Revision 2.0”, 2003. [Online].

Доступно: <https://forums.ni.com/attachments/ni/30/3619/1/LIN.pdf> [приступљено: јул 2019.]

[7] Serial Peripheral Interface. [Online].

Доступно: <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface> [приступљено: јул 2019.]

[8] I²C. [Online].

Доступно: [https://en.wikipedia.org/wiki/I%C2%B2C](https://en.wikipedia.org/wiki/I%C2%B2C%20) [приступљено: јул 2019.]

[9] MSO4000B and DPO4000B Series Digital Phosphor Oscilloscopes User Manual. [Online]

Доступно: [https://download.tek.com/manual/071281002web.pdf](https://download.tek.com/manual/071281002web.pdf%20) [приступљено: јул 2019.]

[10] MSO4000 and DPO4000 Series Digital Phosphor Oscilloscopes Programmer Manual. [Online] Доступно: [https://download.tek.com/manual/077024801web.pdf](https://download.tek.com/manual/077024801web.pdf%20) [приступљено: јул 2019.]