МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет **ФИЗИЧЕСКИЙ**

Кафедра **автоматизации физико-технических исследований**

Направление подготовки **03.03.02 ФИЗИКА**

Образовательная программа: **БАКАЛАВРИАТ**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(проектно-исследовательский формат)

Святкина Виктория Алексеевна

Тема работы **Разработка CAN-bus протокола прикладного уровня для создания распределённых микроконтроллерных систем управления**

«К защите допущена»

И. о. зав. кафедрой

к.т.н., доцент

Лысаков К.Ф. /……..
(фамилия И., О.) / (подпись, МП)

«……»……20…г.

Научный руководитель

д.т.н

зав. лаб. ИАиЭ СО РАН

Зюбин В.Е./……..
(фамилия И., О.) / (подпись, МП)

«……»……20…г.

Дата защиты: «……»……20…г.

Новосибирск, 2024

Оглавление

1. Введение

В последние годы распределенные системы управления стали широко применяться в различных отраслях промышленности, таких как автомобильная, энергетическая, производственная и другие. Одной из ключевых технологий, обеспечивающей эффективное взаимодействие и передачу данных между компонентами распределенной системы, является CAN-bus протокол.

CAN-bus (Controller Area Network) - стандартный протокол, основанный на обмене сообщениями, который широко используется для передачи данных в распределенных системах управления. Он обеспечивает надежную и эффективную связь между различными компонентами системы, такими как датчики, исполнительные механизмы и контроллеры. Этот протокол обладает такими преимуществами, как экономичная проводка, устойчивость к электрическим помехам, самодиагностика и исправление ошибок [1].

Однако действующий стандарт CAN ограничивается спецификацией только двух самых нижних уровней эталонной семиуровневой модели взаимодействия открытых систем OSI/ISO — физического и канального (рис. 1). Но за рамками стандарта остаются решения таких важных при разработке вопросов, как адресация узлов, распределение между ними CAN идентификаторов, интерпретация содержимого фрейма данных, передача данных длиной более 8 байтов и так далее, то есть все то, что обычно рассматривается на более высоких уровнях, вплоть до седьмого прикладного [2]. Поэтому до сих пор создаются спецификации протоколов прикладного уровня.

Рис. 1. Соотношение эталонной модели OSI/ISO и CAN протоколов

Но существующие CAN-bus протоколы прикладного уровня не всегда полностью удовлетворяют требованиям конкретных приложений, особенно в областях со специфическими требованиями, таких как киберфизические системы. В Институте автоматики и электрометрии СО РАН при попытке использовать протокол CAN для построения распределённой микроконтроллерной системы управления с процесс-ориентированной архитектурой выяснилось, что среди разработанных протоколов прикладного уровня отсутствует протокол, который бы удовлетворял специфике процесс-ориентированных распределённых систем.

Таким образом, целью работы является разработка CAN-bus протокола прикладного уровня для создания распределённых микроконтроллерных систем управления с процесс-ориентированной архитектурой программного обеспечения.

Для достижения цели необходимо выполнить следующие задачи:

1. Проанализировать специфику процесс-ориентированных распределённых систем.
2. Сформулировать требования к разрабатываемому протоколу прикладного уровня.

3. Проанализировать существующие протоколы прикладного уровня на базе CAN-bus.
4. Разработать протокол прикладного уровня, учитывающий сформулированные требования.
5. Реализовать разработанный протокол в виде библиотеки для микроконтроллеров ATmega.
6. Экспериментально исследовать протокол на модельной задаче.

Данная дипломная работа имеет важное практическое значение, так как разработанный CAN-bus протокол прикладного уровня может быть применен в любых промышленных системах, где требуется распределенное управление и надежная передача данных. Ожидается, что результаты работы будут использованы в разработке новых систем управления на основе CAN-bus протокола. В дальнейшем, разработанный протокол прикладного уровня может быть стандартизирован и применен в широком масштабе в промышленности.

2. Техническое задание

Данная работа выполнена в Институте автоматики и электрометрии СО РАН для лаборатории киберфизических систем.

В рамках этой работы требуется разработать CAN-bus протокол прикладного уровня для создания распределённых микроконтроллерных систем управления с процесс-ориентированной архитектурой программного обеспечения.

2.1 Анализ специфики процесс-ориентированных распределённых систем

Распределенную систему управления (РСУ, DCS — Distributed Control System) можно определить, как систему, состоящую из множества устройств, разнесенных в пространстве, каждое из которых не зависит от остальных, но взаимодействует с ними для выполнения общей задачи. Под «множеством

устройств», разнесенных в пространстве, понимается любое микропроцессорное устройство: контроллер, компьютер, модуль ввода-вывода и т.д. [3].

Иначе говоря, распределённая система представляет собой совокупность независимых устройств, не имеющих общей памяти и единого таймера. Эти устройства взаимодействуют друг с другом через сеть, обмениваясь сообщениями. У каждого устройства есть своя собственная оперативная память и операционная система, но несмотря на это, операционные системы взаимодействуют, предоставляя свои ресурсы друг другу для совместного решения общей задачи.

Переход на распределенные системы управления позволяет существенно сократить общие затраты и расходы на обслуживание. Это достигается благодаря использованию компактных, экономичных и гибких микропроцессоров, сокращению длины проводов, упрощению подключения и улучшению ремонтопригодности. Кроме того, такие системы предоставляют возможность внедрения разнообразных стратегий управления.

Процессно-ориентированная парадигма предполагает, что управляющая программа, состоящая из набора слабо связанных параллельных процессов, структурно и функционально соответствует технологическому описанию системы [4]. В рамках процессно-ориентированного программирования программы разрабатываются как совокупность взаимодействующих процессов. Один процесс может запускать другой и отслеживать его выполнение. Каждый процесс определяется последовательностью состояний, которые задаются с помощью арифметических конструкций и дополнены операциями тайм-аута, установки состояния, а также командами запуска, остановки и проверки состояния для взаимодействия с другими процессами.

Предварительные исследования возможности применения процесс-ориентированной парадигмы программирования для спецификации управляющих алгоритмов, реализуемых на распределенных микроконтроллерных архитектурах, показали обнадеживающие результаты,

которые заключаются в возможности топологически независимого описания алгоритма управления и существенного снижения загрузки сети на межузловой обмен [5, 6].

Основываясь на сказанном выше, была выявлена специфика процесс-ориентированных распределённых систем. Во-первых, необходимо обеспечить возможность передавать большие данные на высокой скорости. Во-вторых, необходима поддержка 100 и более узлов в сети. В-третьих, подобная система предполагает минимизацию накладных расходов на отсеивание принятых сообщений. В-четвёртых, необходима поддержка межпроцессного взаимодействия и возможность анализа исходных кодов, чтобы убедиться в отсутствии уязвимостей и неприемлемых для пользователя функций.

Исходя из этой специфики были сформулированы требования к разрабатываемому протоколу прикладного уровня:

1. скорость передачи данных – от 1 Мбит/с,
2. ограничение на длину пакета данных – от 1 Кбайт,
3. наличие механизма фильтрации сообщений,
4. ограничение на число узлов в сети – до 100,
5. возможность удалённой загрузки,
6. p2p передача данных,
7. поддержка межпроцессного взаимодействия,
8. возможность анализа исходных кодов.

2.2 Анализ существующих протоколов прикладного уровня на базе CAN-bus

В ходе работы были проанализированы протоколы прикладного уровня на базе CAN-bus на соответствие сформулированным ранее требованиям. Были проанализированы протоколы CANopen, Can Kingdom, DeviceNet, SDS, ARINC 825, EnergyBus, IEC 61375-3-3, ISOBUS, ISO-TP, MilCAN, NMEA

2000, SAE J1939, Unified Diagnostic Services (UDS), LeisureCAN и соответствующая документация. Результаты приведены в таблице 1.

	Максимальная длина сообщений, байт	Открытость	Допустимые номера узлов	Допустимые скорости передачи данных, кбит/с	Возможность удалённой загрузки	p2p взаимодействие
Требования	1000	Да	Не менее 100	1000	Да	Да
CANopen	64	Да	0-127	10, 20, 50, 125, 250, 500, 800, 1000	Да	Да
Can Kingdom	Определяется реализацией	Нет	0-255	Любые до 1000	Да	Нет
DeviceNet	8, если больше, фрагментация	Нет	0-63	125, 250, 500	Нет	Да
SDS	6, если больше, фрагментация	Нет	0-125	125, 250, 500, 1000	Да	Нет
ARINC 825	64	Да	1-254	10-1000	Да	Да
EnergyBus	8, если больше, фрагментация	Да	0-127	250	Да	Нет
IEC 61375-3-3	64	Да	1-127	500, 1500	Да	Нет
ISOBUS	8, если больше, фрагментация	Да	1-254	250, 500	Да	Нет
ISO-TP	2^32 (с помощью фрагментации)	Да	0-2047	0-1000	Нет	Нет
MilCAN	64	Нет	0-127	0-1000	Нет	Да
NMEA 2000	8	Нет	0-254	250	Да	Да
SAE J1939	8, если больше, фрагментация	Да	0-253	250, 500	Нет	Нет
Unified Diagnostic Services (UDS)	7, если больше, фрагментация (до 4095)	Да	1-253	10-1000	Да	Нет
LeisureCAN	8	Да	1-127	125, 250, 500	Да	Нет

Таблица 1. Результаты анализа протоколов прикладного уровня на базе CAN-bus

Таким образом, анализ существующих протоколов показывает, что для целей разработки распределенных микроконтроллерных систем на основе процесс-ориентированной парадигмы программирования требуется создание специализированного протокола прикладного уровня.

3. Предлагаемый протокол

Система представлена коммуникационным узлом и периферийными модулями. На идентификацию сообщения отводится 11 бит поля арбитража (исключая бит запроса удалённой передачи RTR).

Семь бит (ID0 – ID6) представляют собой идентификатор периферийного модуля. Идентификаторы принимают значения от 0x001 до 0x07D, таким образом максимальное количество периферийных узлов в системе 125 (рис. 2).

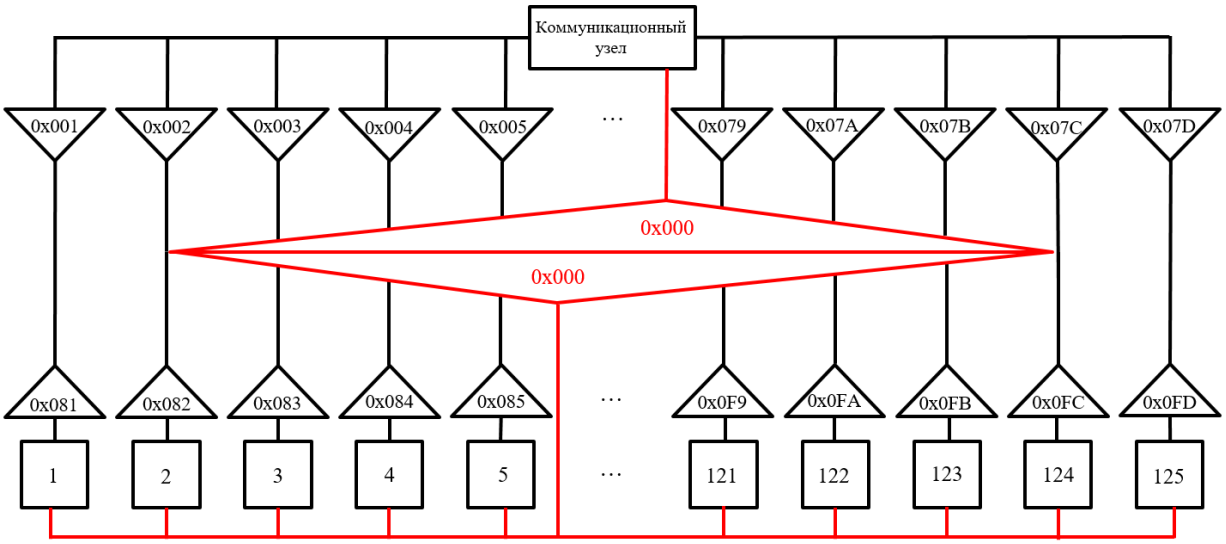

Рис. 2. Предлагаемая система с идентификаторами сообщений

Восьмой бит (ID7) отведён для идентификации направления сообщения (рис. 3). Если сообщение отправляет коммуникационный узел, этот бит равен 0, а если периферийный, то 1.

Рис.3. Значения бит поля арбитража в предлагаемом протоколе

Также выделены идентификаторы для целей: широковещательного сообщения (0x000), идентификаторы, используемые при удалённой загрузке

модулей (0x07E, 0x0FE) и идентификаторы неинициализированного модуля (0x07F, 0x0FF).

3.1 Механизм маскирования

Предлагаемый протокол реализует механизм маскирования. Фильтры и маски используются для того, чтобы определить нужно ли принять или отбросить пришедшее сообщение. Если сообщение считается допустимым, то оно будет загружено в один из двух буферов приема.

Как только сообщение пришло, поля идентификатора пришедшего сообщения сравниваются со значениями фильтра. Если обнаружено соответствие (то есть обнаружено допустимое сообщение), то это сообщение будет загружено в один из буферов приема [7].

Маска используются, чтобы определить, какие биты в идентификаторе должны совпадать с битами фильтров. Иначе говоря, маска определяет, какой из битов фильтра используется для фильтрации. Если любой бит в маске установлен в 0, то это означает, что ни один из битов фильтра не применяется, так что сообщения будет автоматически принято, независимо от состояния битов фильтра (таблица 2).

Бит маски n	Бит фильтра n	Бит ID сообщения	Принять или отбросить сообщение по биту n
0	X	X	Принять
1	0	0	Принять
1	0	1	Отбросить
1	1	0	Отбросить
1	1	1	Принять

Таблица 2. Принцип работы механизма фильтрации

В предлагаемом протоколе узлы задействуют основной буфер приёма сообщений. За счёт использования фильтрации периферийные узлы принимают широковещательные сообщения и сообщения со своим

идентификатором в поле идентификатора сообщения. То есть значение маски для любого периферийного узла составляет 0x0FF, значение первого фильтра 0x000, а второго 0x000 + идентификатор узла (рис. 3). Коммуникационный узел не использует фильтрацию.

3.2 Межузловое взаимодействие

Биты ID8 – ID10 поля арбитража предлагается использовать для общения периферийных узлов между собой. Соответственно, каждой паре отправитель сообщения – получатель присваивается номер связи (от 1 до 7).

Отправитель, посылая сообщение, заполняет биты ID8 – ID10 идентификатором связи, идентификатор направления (ID7) ставит в 0, в остальные биты поля арбитража (ID0 – ID6) записывается идентификатор получателя сообщения.

Для ответа получатель заменяет собственный идентификатор на идентификатор отправителя в битах ID0 – ID6 и отправляет сообщение (рис. 4).

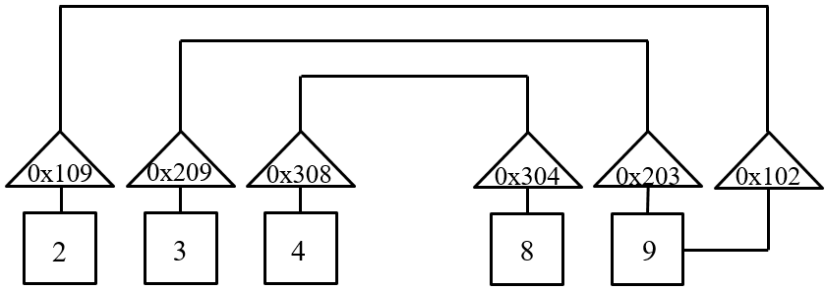

Рис. 4. Идентификаторы сообщений при общении периферийных узлов

Для определения идентификаторов межузловых связей был разработан следующий алгоритм:

1. Построение графа связей периферийных узлов.
2. Обход узлов.
3. Поиск пронумерованных ранее связей для текущего узла.
4. Определение для рассматриваемой связи свободного идентификатора.
5. Присвоение связи свободного идентификатора.

6. Переход к следующему узлу.

Блок схема алгоритма представлена на рис. 6. Алгоритм реализован на языке Си.

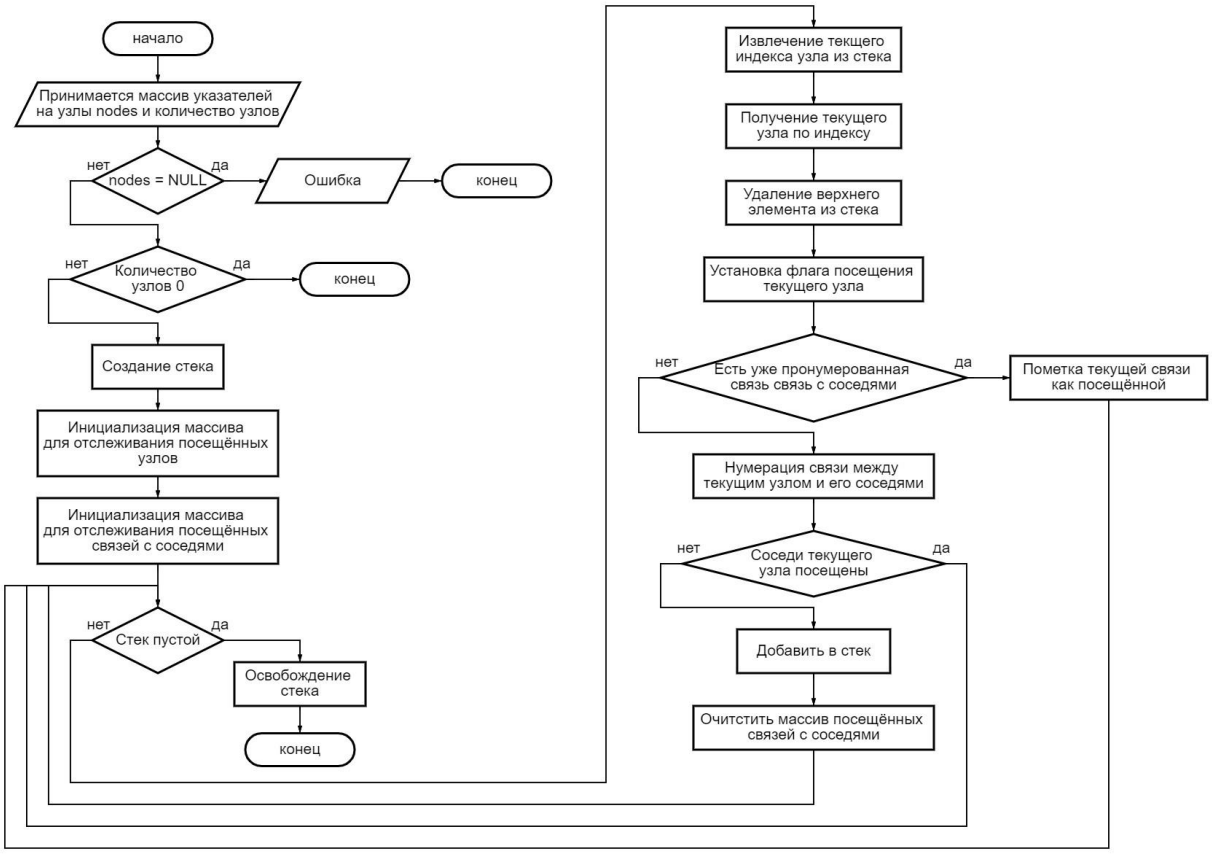

Рис.6. Блок схема алгоритма определения идентификаторов межузловых связей

3.3 Удалённый запуск процессов

Удалённый запуск процессов предлагается реализовать следующим образом:

1. Узел, запрашивающий выполнение процесса (Source):
    1) Установка флага в локальной копии слова-состояния процесса.
    2) Посылка CAN-сообщения с командой запуска процесса.
    3) Получение CAN-сообщения о завершении процесса.
    4) Снятие флага в локальной копии слова-состояния процесса.
2. Узел, выполняющий процесс (Destination):
    1) Получение CAN-сообщения с командой запуска процесса.

2) Сохранение адреса инициатора запуска (по идентификатору связи).
3) Запуск процесса.
4) Завершение процесса.
5) Посылка CAN-сообщения о завершении процесса узлу, запрашивавшему выполнение процесса.

Диаграмма последовательности представлена на рис. 7.

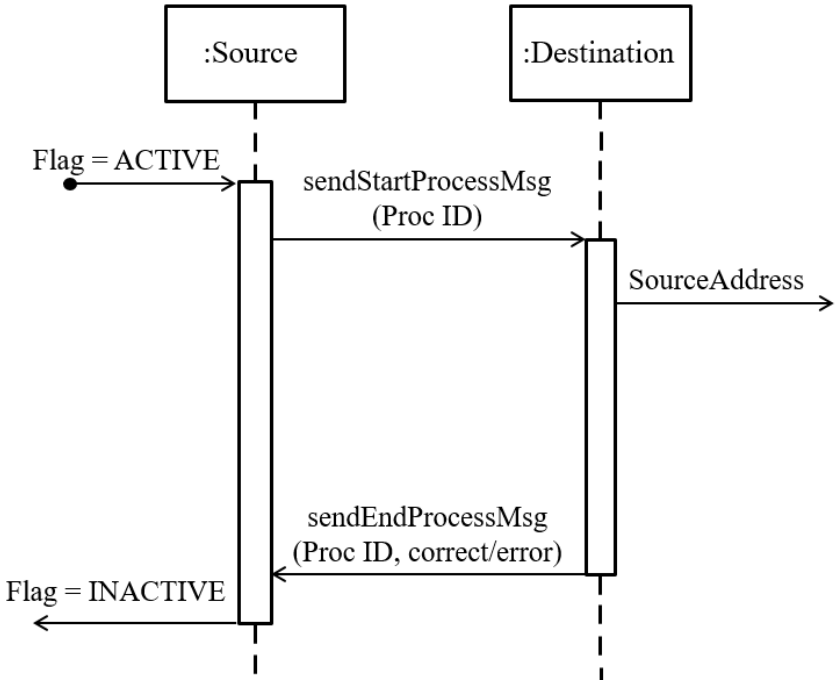

Рис. 7. Диаграмма последовательности удалённого запуска процессов

3.4 Реализация протокола и его эмпирическое исследование

Протокол реализован, как надстройка над библиотекой mcp_can.h [8]. Реализован макет из двух микроконтроллеров Arduino Uno семейства ATmega и двух контроллеров шины CAN MCP2515 (рис. 8).

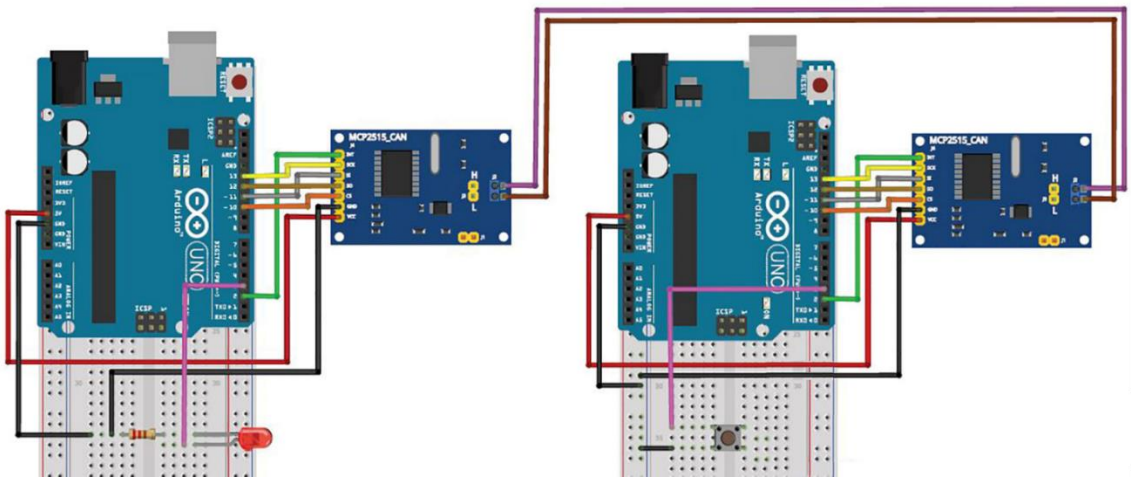

Рис. 8. Макет

Протестирована работа механизмов фильтрации сообщений, передачи длинных пакетов данных и удалённого запуска процессов.

Эмпирическое исследование разработанного протокола прикладного уровня на базе CAN-bus было проведено на имитаторе модуля управления тиристорами для распределённой системы управления установкой анодного оксидирования (Рис. 9).

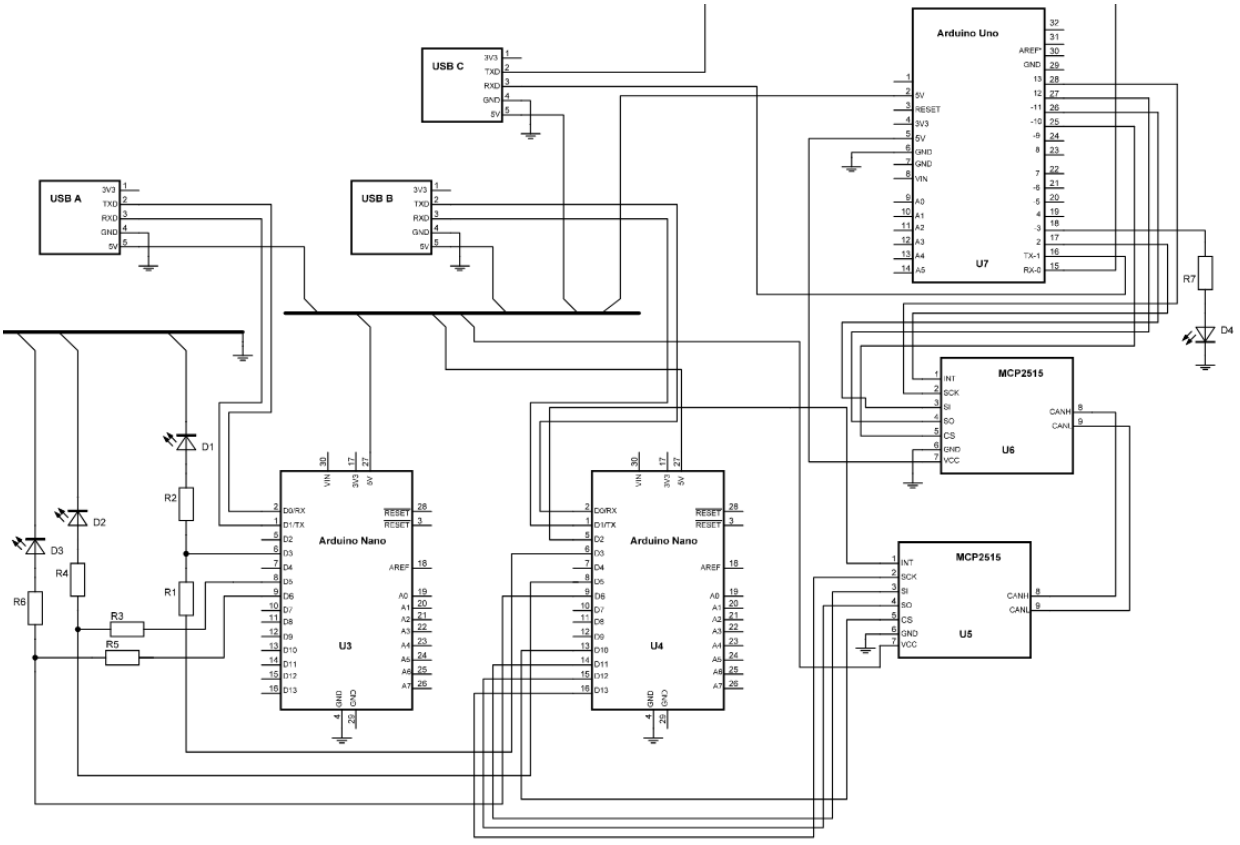

Рис. 9. Схема имитационного стенда

4. Заключение

Разработанный CAN протокол прикладного уровня, который специфицирует алгоритмы идентификации узлов и межузловых связей, удалённого запуска процессов, упрощает разработку распределённых микроконтроллерных систем управления на основе процесс-ориентированной парадигмы программирования.

Разработанный протокол является надстройкой над библиотекой mcp_can.h, позволяет подключить 125 периферийных узлов, реализует возможность подключения не инициализированных модулей на загрузку и передачи длинных сообщений (1 Кбайт) на высокой скорости (1 Мбит/с).

В настоящее время идёт интегрирование протокола в Си-подобный процесс-ориентированный язык Рефлекс.

Список литературы

1. Bozdal M. et al. Evaluation of can bus security challenges //Sensors. – 2020. – Т. 20. – №. 8. – С. 2364
2. Щербаков А. Протоколы прикладного уровня CAN-сетей //Современные технологии автоматизации. – 1999. – №. 3. – С. 6-15
3. Романов А. Распределенные вычисления и приложения: учебное пособие – Ульяновск : УлГТУ, 2018. – 151 с.
4. V. E. Zyubin, A. S. Rozov, I. S. Anureev, N. O. Garanina and V. Vyatkin, "poST: A Process-Oriented Extension of the IEC 61131-3 Structured Text Language," in IEEE Access, doi: 10.1109/ACCESS.2022.3157601.
5. Zyubin, V.E.; Garanina, N.O.; Anureev, I.S.; Staroletov, S.M. Towards Topology-Free Programming for Cyber-Physical Systems with Process-Oriented Paradigm. Sensors 2023, 23, 6216.
6. Зюбин В.Е. Реализация распределенной системы управления с сохранением семантики гиперпроцесса // Сборник трудов Российской конференции с международным участием «Распределенные информационно-вычислительные ресурсы (DICR-2022)», Россия, г. Новосибирск, 5 – 8 декабря 2022 г. С. 91-97 DOI: 10.25743/DIR.2022.51.99.016
7. Stand-Alone CAN Controller with SPI Interface [Электронный ресурс]. – URL: https://www.microchip.com

8. Cory J. MCP_CAN_lib [Электронный ресурс]. – URL: https://github.com/coryjfowler/MCP_CAN_lib