DOI 10.36622/VSTU.2022.25.3.012
УДК 004.056.55/003.26

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ КИБЕРФИЗИЧЕСКИХ СИСТЕМ

П.Ю. Филяк, И.А. Тырин, Д.А. Пажинцев

Активное внедрение информационных технологий, таких как облачное хранилище или интернет, электронная коммерция, в современном мире создаёт обширные предпосылки к увеличению числа «умных» устройств, а, следовательно, и формированию различных киберфизических систем (CPS). Такие системы уже активно используются в различных сферах жизни: медицина, промышленность, военные структуры, КИИ и т.д. Ввиду того, что концепция состоит из множества технологических тенденций, для рассмотрения было выбрано одно из ключевых направлений, в частности, киберфизические системы. Актуальность выбора этого сегмента и в целом темы исследований обосновывается такими факторами, как стремительный рост числа «умных» устройств, вследствие чего увеличивается число воздействий и атак на устройства и системы в целом. Поэтому информационная безопасность CPS на сегодня является одним из самых востребованных направлений изучения и исследований.

Ключевые слова: киберфизические системы, CPS, информационная безопасность, интерфейс, протоколы, каналы передачи данных, защищенные информационные системы, каналы управления перехват, злоумышленники.

Киберфизическую платформу можно разделить на три составляющие, а именно: интернет людей, интернет вещей и интернет-сервисов.

Интернет людей – это тот период времени, когда номинально число пользователей Интернета не превышало количество населения Земли.

«Интернет людей» закончился между 2008 и 2009 годами, и начался «Интернет вещей».

Под интернетом вещей понимается, прежде всего, идея вычислительной сети, которая соединяет в себе физические предметы, которые в свою очередь оснащены датчиками и сенсорами для взаимодействия между собой или с внешней средой [1].

Такое взаимодействие происходит с помощью сети интернет, а именно IP-адреса и протокола управления передачей – TCP. Стоит понимать, что уже существует огромное количество умных вещей, а, следовательно, для их взаимодействия друг с другом, адресов в протоколе TCP/IPv4 будет недостаточно.

Большим толчком в развитии Интерната вещей стало появление нового сетевого протокола IPv6, который во много раз увеличивает количество подключений в сети.

Такая сеть связывает между собой огромное множество устройств, которые способны отслеживать состояние внешней среды, собирать информацию, а также взаимодействовать с любым другим таким же устройством.

Интернет-вещи работают с помощью Интернет-сервисов, туда, в том числе входят приложения для управления Интернет-вещами. Нельзя забывать, что те самые приложения – в первую очередь программы, написанные на определённом языке программирования определёнными людьми. Учитывая человеческий субъективизм, а также актуальные для человеческой психики и в общем человека угрозы, неизбежны ошибки в человеческой деятельности, в данном случае – в написанных им приложениях. Схематично такое определение представлено на рис. 1.

В такой системе, способной к взаимодействию с другими системами или подсистемами без каких-либо ограничений, входящие в неё компоненты (подсистемы) (рис. 2) работают по независимым наборам команд, и управляются через единый профиль [2].

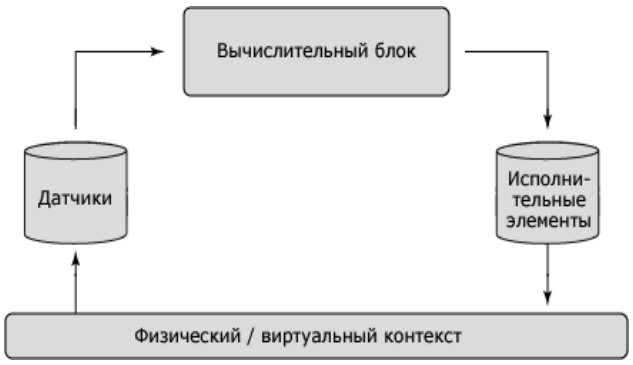

Рис. 1. Структура киберфизической системы

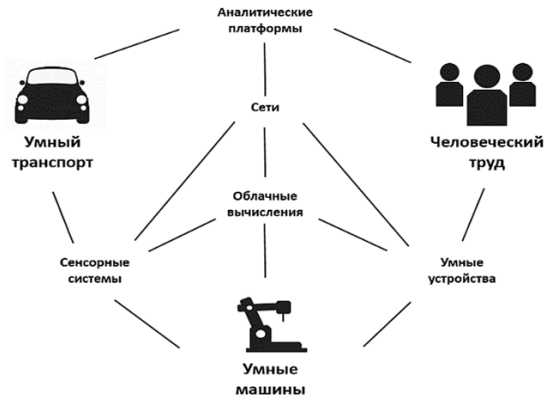

Рис. 2. Составляющие киберфизических систем

Особенности киберфизических систем. Промышленный интернет вещей

Через некоторое время после своего зарождения, IoT постепенно начал внедряться в промышленность, что дало скачек зарождению такого понятия, как Промышленный интернет вещей, вследствие чего увеличивается эффективность работы предприятия [3-5] (рис. 3).

Рис. 3. Промышленный интернет вещей

Современные примеры киберфизических систем и систем промышленного интернета вещей

При помощи CPS людям становится жить на много проще и комфортнее. Подтвердить данный факт можно на примере «умных» городов. Ни для кого не секрет, что Сингапур признается самым «умным» городом. Также в Сингапуре продолжает развивается Smart Nation – этот проект, Разработчики данного проекта сделать жизнь горожан более простой и комфортной.

Не исключением становится и Сыктывкар, где активно внедряется проект «Безопасный город», который является частью проекта «Умный город». В рамках данного проекта на территории города устанавливают камеры видео наблюдения, в том числе камеры с функцией видеоаналитики, доступ к которым предоставляется правоохранительным органам. Также в рамках проекта строят рубежи системы автоматической фиксации нарушений ПДД с передачей доступа правоохранительным органам к данным для осуществления оперативно-разыскных мероприятий [6-8].

Также ярким примером использования промышленного интернета вещей является распределительный центр Amazon. Распределительный центр Amazon в Аризоне занимает площадь около 28 футбольных полей. На объекте с такой огромной площадью становится затруднительно искать нужные товары, упаковывать их и отправлять клиентам. Человек на территории с такой площадью может попросту потеряться.

Чтобы уменьшить число сотрудников на

складе, используются дроны. Сейчас дроны используются в 13 центрах Amazon по всему миру, что уже снизило расходы компании на 20% [9].

Примеры кибератак на киберфизические системы и системы промышленного интернета вещей

Атака с помощью зловреда Stuxnet самых известных кибератак, в результате которой из строя были выведены иранские центрифуги по обогащению урана, тем самым иранская ядерная программа была отброшена на несколько лет назад.

В декабре 2015 в городе Иваново-Франковск группа хаккеров провела атаку на электрические подстанции, тем самым из строя было выведено 30 подстанций. В результате 230 тысяч человек остались без электричества.

Также отмечается, что перед отключением электричества, когда оператор в компании «Прикарпатьеоблэнерго», попытался это остановить, его выкинуло из системы [10].

В марте текущего года группа хаккеров провела атаку на сервис компании Verkada, который предоставляет услуги корпоративного видеонаблюдения. В результате проведенной атаки злоумышленники получили доступ к более чем 150 тысячам камер видеонаблюдения. Один из самых крупных клиентов Verkada - компания Tesla, к камерам с заводов и складов которой злоумышленники получили доступ [10].

Анализ платформ для управления смарт-устройствами и выбор подходящей для реализации сети IIOT

На сегодняшний день существует огромное множество современных и серьёзных IIoT – решений. Так лидирующие позиции на Российском рынке среди поставщиков готовых решений под Промышленный Интернет вещей занимают: Qmodule, Тингеникс, ИНФОТЕХ ГРУПП, ЦИФРА.

Стоимость большинства современных и серьёзных решений достаточно велика, а демонстрационных версий таких систем попросту нет, поэтому будут рассмотрены наиболее популярные платформы, использующиеся в Интернете вещей: iRidium, MajorDoMo, OpenHAB.

iRidium

iRidium – комплекс программ для управления оборудованием домашней и коммерческой автоматизации, безопасностью и мультимедиа. Позволяет работать с разным оборудованием автоматизации через один пользовательский интерфейс [11].

iRidium управляет всем оборудованием, которое включено в систему автоматизации. Обычно, iRidium используют проектировщики систем автоматизации, т.к. создание пульта возможно только при знании архитектуры системы.

Одним из самых важных преимуществ iRidium является наличие такого компонента, как iRidium server. iRidium server имеет как программную, так и аппаратную реализацию. В аппаратной реализации имеется выход KNX, что является её отличительной особенностью.

iRidium предполагает два варианта реализации проекта «умного» дома – с сервером и без него. Однако, при реализации проекта без сервера вы можете столкнуться с такой проблемой, что система будет плохо воспринимать большое количество пультов управления, поэтому вы не сможете поставить в каждой комнате управляющую панель. Вам придется носить ее с собой постоянно, а это не совсем удобно. Сеть с сервером обходится дороже, но у нее нет ограничений по количеству управляющих панелей (рис. 4). Вы сможете разместить управляющие панели на базе смартфонов или планшетов в каждой комнате и в полной мере использовать весь потенциал данной платформы [12].

Еще одним мощным компонентом, который выделяет iRidium среди других аналогичных платформ, является iRidium Studio. iRidium Studio – это среда для разработки интерфейсов и серверных проектов. iRidium Studio позволяет создать красивый интерфейс в зависимости от ваших пожеланий (рис. 5).

Рис. 4. Аппаратная реализация iRidium server

Рис. 5. Интерфейс iRidium Studio

MajorDoMo

MajorDoMo (Major Domestic Module или Главный домашний модуль) - платформа для комплексного управления домашней автоматикой, а также для информационной поддержки жизнедеятельности, находящаяся в свободном доступе (рис. 6).

MajorDoMo предоставляет единый интерфейс для оборудования различных типов.

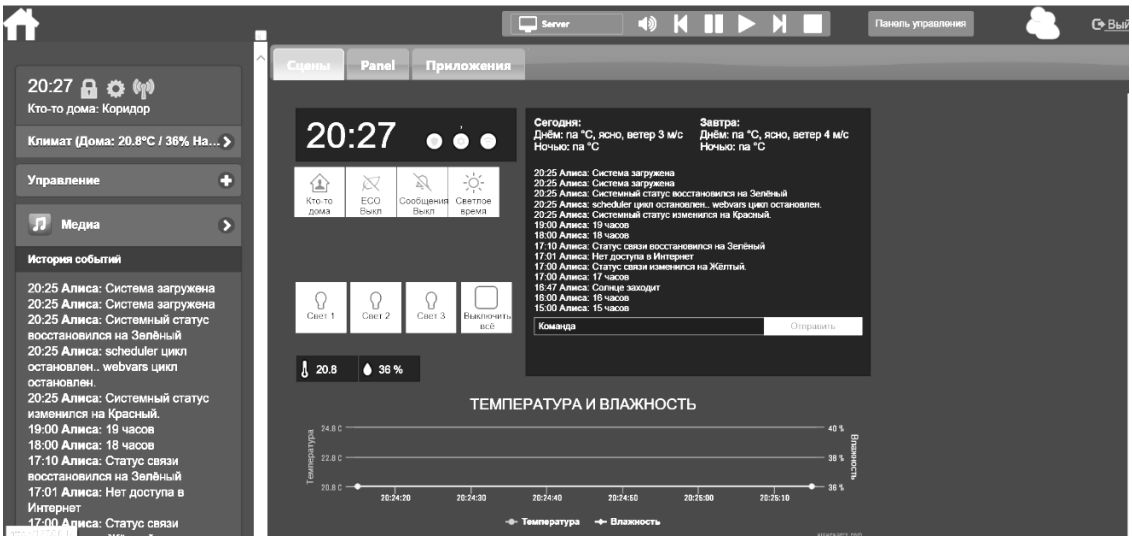

Рис. 6. Главная страница MajorDoMo

У вас также есть возможность создания собственных сценариев, путем написания скриптов на PHP. Обращение к устройствам будет производиться в рамках его синтаксиса и их объектных имен. В том случае, если вы не владеете навыками программирования, то в MajorDoMo реализована функция создания своих сценариев при помощи внутреннего конструктора Blockly - инструмента «визуального программирования» (рис. 7). Вместо написания кодовых последовательностей применяются их графические представления. Для построения алгоритма достаточно собрать их в необходимой последовательности (рис. 8).

```
say("Доброе утро!");
//Если дома гости, то не нужно включать музыку
if (!getGlobal('ThisComputer.WeHaveGuests')) {
 runScript("sayTodayAgenda");
 runScript("playFavoriteMusic");
}
```

Рис. 7. Скрипт сценария, написанный на PHP

Рис. 8. Скрипт сценария, составленный с помощью Blockly

Еще одна очень полезная функция, реализованная в MajorDoMo, это парсинг информации со сторонних web-страниц. Взятые из сети данные можно использовать в создании новых сценариев. Приведем следующий пример: на синоптическом сайте по прогнозу наблюдается резкое похолодание или потепление. Мы можем настроить MajorDoMo, чтобы в зависимости от этих данных он давал команду на увеличение или понижение температуры в доме для того, чтобы в доме был комфортный микроклимат, с учетом внешней температуры.

OpenHAB

OpenHAB (Open Home Automation Bus) нацелен на создание универсальной платформы для объединения всех «умных» вещей в вашем доме в единую систему управления [13]. Одно из первых открытых решений для «умного» дома (рис. 9). Можно сказать, что OpenHAB стал прообразом для всего разнообразия платформ, сейчас мы можем наблюдать OpenHAB, что может быть любой сложности [13].

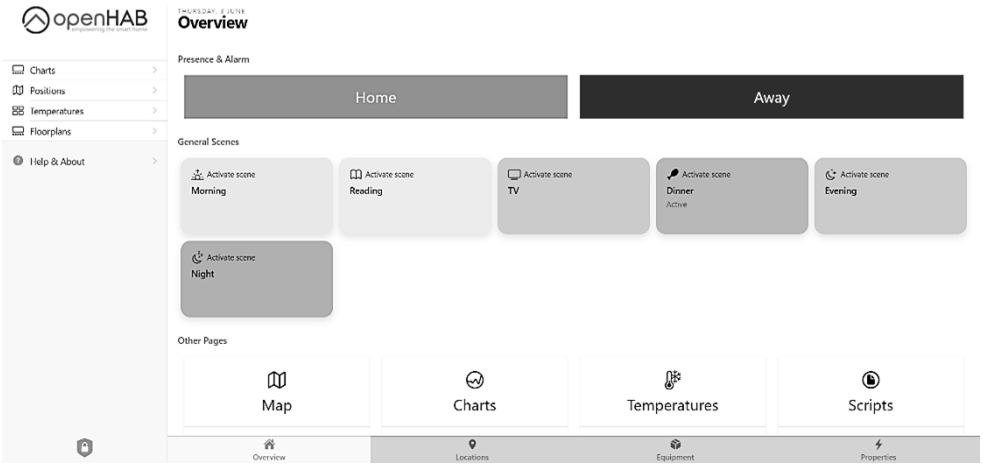

Рис. 9. Главная страница OpenHAB

Как и другие платформы OpenHAB предоставляет вам возможность самостоятельно разработать управляющую панель так, как вы этого захотите. Также вы самостоятельно можете создавать сценарии и правила, регулирующие работу устройств. Сама платформа основана на Java, но в работе с OpenHAB используется собственный скриптовый язык для автоматизации и сценариев - DSL (Domain Specific Language). Перейдя на вкладку «Code» на главной странице OpenHAB, вы можете самостоятельно написать необходимый вам код, для создания дизайна главной страницы или же для создания необходимых для вас сценариев работы устройств (рис. 10).

В случае, если вы не владеете навыками программирования, вы также без труда сможете произвести необходимые настройки, перейдя во вкладку «Design». Все ваши настройки автоматически конвертируются в код, который вы можете просмотреть во вкладке «Code» (рис. 11).

Результаты сравнения

Одной из главных задач ВКР является анализ и последующий выбор платформы для добавления её в создаваемую сеть IIoT. Изучив каждую из вышеперечисленных платформ, видим, что каждая из них хороша по-своему.

Рис. 10. Вкладка «Code»

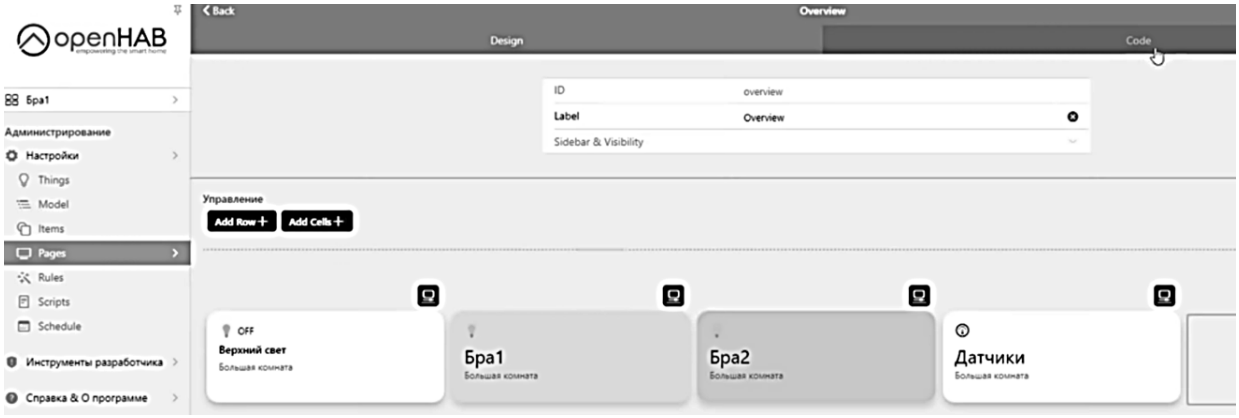

Рис. 11. Вкладка «Design»

С помощью iRidium можно отлично реализовать более масштабные проекты, нежели «умный» дом, а именно: «умный» офис, «умный» жилой комплекс и т.д. Что касается MajorDoMo, у данной платформы огромное русскоязычное сообщество, что существенно упрощает работу. Однако для реализации в сети был выбран OpenHAB, ввиду следующих особенностей:

- программное обеспечение, распространяемое бесплатно;
- возможность реализации безопасного подключения к MQTT-брокеру при помощи использования SSL/TLS;
- более современный и презентабельный интерфейс по сравнению с MajorDoMo;
- наличие механизма безопасности SSL-pinning.

Анализ существующих прикладных протоколов, использующихся в интернете вещей

Типовая, а точнее её архитектура состоит из следующих взаимодействующих между собой уровней: сенсорные узлы и сеть датчиков, сервера, приложения. Ввиду того, что нижний уровень сети предполагает собой наличие датчиков и узлов, возникает острая нужда в новых «уникальных» прикладных протоколах для взаимодействия датчиков как между собой, так и с верхними уровнями. Такие контроллеры обычно представляют из себя малогабаритные устройства с маленькой памятью и низким энергопотреблением, которые измеряют различные параметры в режиме реального времени, после чего отправляю их на сервер.

Шаблоны взаимодействия для интернета вещей

Прежде чем проводить анализ существующих протокол использующихся в сетях Интернета вещей, необходимо рассказать о существующих шаблонах взаимодействия в IoT-системах. Существует не малое количество шаблонов, но среди них выделяются два основных - «запрос - ответ» и «издатель - подписчик». Наиболее популярный протокол, который поддерживает шаблон «запрос – ответ», является HTTP. Кроме него, шаблон поддерживают протоколы XMPP и CoAP, которые будут упомянуты далее.

Шаблон «издатель-подписчик»

IШаблон «издатель – подписчик» будет наиболее эффективным способ обмена данными между контроллерами в IoT-сети. Главным преимуществом этой модели взаимодействия является разделение получающей и отправляющей стороны, когда как в системе они обе рассматриваются как клиенты. Так клиент, являющийся издателем, оправляет информацию единожды брокеру сообщений, представляющим из себя сервер посредник. что существенно уменьшает трафик, позволяя повысить уровень масштабируемости, так необходимый для гигантских IoT-сетей.

Дополнительным преимуществом будет то, что обычно брокеры могут предоставлять службу аутентификации для своих клиентов, таким образом облегчая работу в сети.

Этот шаблон взаимодействия поддерживают такие прикладные протоколы, как XMPP, MQTT, AMQP, о которых также будет упомянуто ниже.

Прикладной уровень	MQTT	CoAP	XMPP	Другие
Транспортный уровень	TCP		UDP	
Сетевой уровень	IPv4		IPv6+6LoWPAN	
Уровень доступа к сети	IEEE 802.3	IEEE 802.11	IEEE 802.15	IEEE 802.16

Рис. 12. Стек протоколов для Интернета вещей

Анализ прикладных протоколов в сети Интернета вещей

Контроллеры и датчики должны взаимодействовать друг с другом. Затем, после сбора данных, их нужно передать в серверную инфраструктуру, которая в свою очередь должна совместно использовать полученные данные и иметь возможность отправлять данные обратно контроллерам и приложениям для анализа (рис. 12). В качестве отправной точки была взята следующая топология сети Интернета вещей (рис. 13).

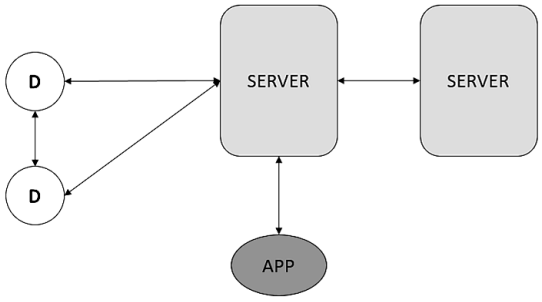

Рис. 13. Взятая в рассмотрение топология сети Интернета вещей

В настоящий момент в сети IoT широко применяются следующие прикладные, стандартизированные протоколы:

- MQTT (протокол, применяемый для сбора данных с контроллеров и датчиков и дальнейшей отправки их серверам);
- XMPP (частный случай взаимодействия между контроллерами и серверами);
- DDS (протокол для взаимодействия датчиков и контроллеров между собой);
- CoAP (протокол, применяемый при взаимодействии клиентов и серверов, реализующий шаблон «запрос - ответ»);
- SOAP Каждый из этих протоколов широко распространён. Есть по крайней мере 10 вариантов реализации каждого из них.

Идеально подходит для систем, где необходим маленький размер программного кода и есть наличие ограничений по пропускной способности канала. Всё это делает протокол идеальным решением для систем с межмашинным взаимодействием и систем IIoT.

MQTT работает поверх протокола TCP, Для подключения к брокеру сообщений используется по умолчанию порт 1883 (при

установке зашифрованного соединения TLS/SSL используется порт 8883).

Взаимодействие при обмене сообщениями в MQTT происходит между клиентами сети и брокером сообщений, стоит отметить, что клиент может выступать как в качестве издателя, так и в качестве подписчика.

Топология сети, которая использует протокол MQTT выглядит иначе, чем общая топология, приведённая выше, ввиду использования шаблона «издатель-подписчик» и как следствие наличия промежуточного узла (брокера) между датчиками и сервером (рис. 14). Важно отметить, что и датчики, и сервер в данном случае будут являться клиентами такой сети. Публикация сообщений происходит путём отправки данных брокеру сообщений, с указанием определённого топика, при помощи которого другие клиенты смогут получить опубликованное сообщение. Клиенты могут быть как подписчиками, так и издателем (рис. 15).

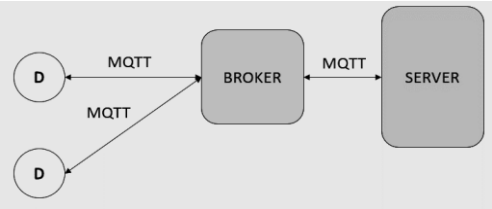

Рис. 14. Топология сети Интернета вещей с брокером

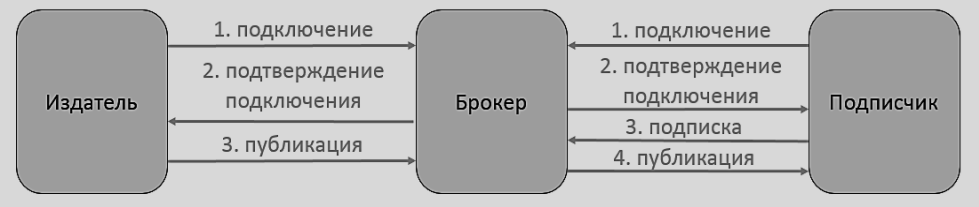

Рис. 15. Схема взаимодействия сторон в шаблоне "издатель-подписчик"

Нулевой уровень. Клиент публикует сообщение с QoS = 0 и в дальнейшем не ждёт подтверждения от брокера о получении сообщения (рис. 16).

Рис. 16. Нулевой уровень QoS

Первый уровень («минимум один раз»). Когда сообщение публикуется с QoS =1, брокер подтверждает получение, обеспечивая гарантированную доставку, которую подтверждает получатель (рис. 17).

Рис. 17. Первый уровень QoS

Второй уровень («ровно один раз»). Сообщения отправляются ровно один раз и без дублирования (рис. 18) [14-16].

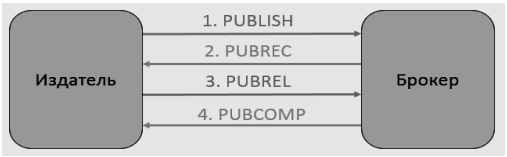

Рис. 18. Второй уровень QoS

Data Distribution Service

DDS (Data Distribution Service) – протокол для обеспечения связи между контроллерами и устройствами, при помощи реляционной модели данных. На участке сети, изображенном на рис. 19, происходит обмен данными.

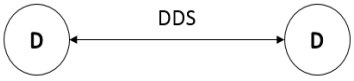

Рис. 19. Протокол DDS на участке между датчиками

В отличие от протокола MQTT, который использует шаблон «издатель-подписчик», XMPP работает поверх транспортного протокола TCP (есть возможность создания защищённого соединения при помощи криптографических протоколов TLS/SSL), использует текстовый формат XML для сообщений и поддерживает различные

шаблоны взаимодействия, такие как: «запрос – ответ», «издатель – подписчик» и другие.

Протокол не поддерживает качество обслуживания QoS, что делает его применение в сетях Интернета вещей непрактичным. Однако такая адресация бывает удобна при передаче данных между отдалёнными точками. К можно в домашнем использовании можно создать собственный XMPP сервер на подобии почтового, а дальше подключить к нему домашние контроллеры, устройства для взаимодействия с ними издалека (рис. 20).

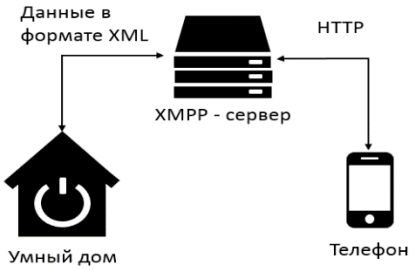

Рис. 20. Частная реализация сети с использованием XMPP

В сетях, которые отличаются низким энергопотреблением используется шаблон взаимодействия «запрос – ответ» (рис. 21).

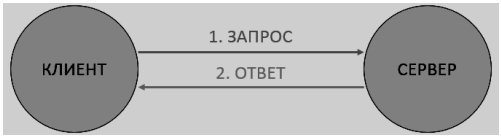

Рис. 21. Взаимодействие сторон в шаблоне "запрос-ответ"

Эта концепция предусматривает наличие клиента, который обращается к серверу, посылая различные команды, и сервера. По посылаемым командам, можно понять, что CoAP аналогичен HTTP, вследствие чего его можно рассматривать как дополнение к протоку передачи гипертекста. CoAP использует в качестве транспортного протокола UDP, обеспечивая тем самым уменьшение требований к пропускной способности сети.

Протокол нашёл своё применение в решениях, когда система управления контроллерами интегрируется в веб-приложение и так далее [15].

Результаты аналитического обзора прикладных протоколов

Результаты анализа упомянутых ранее прикладных протоколов взаимодействия в Интернете вещей представлены в табл. 1, в которой протоколы классифицируются по назначению, особенностям и используемому транспортному протоколу.

Таблица 1

Результаты анализа

Протокол	Трансп. протокол	Назначение	Особенности
XMPP	TCP		
DDS	UDP	Для сетей, нуждающихся в распределении нагрузки	Распределение данных между устройствами, через прямую шину
MQTT	TCP	и брокером	Поддержка качества обслуживания QoS и наличие брокера сообщений
CoAP	UDP		Учитывает различные вопросы реализации в ограниченных средах
SOAP	TCP	Для распределительной вычислительной сети	

На основании приведённого анализа в качестве прикладного протокола для практической реализации сети Интернета вещей был выбран протокол MQTT ввиду следующих преимуществ:

- протокол работает поверх TCP;
- поддерживает качество обслуживания QoS, обеспечивая проверку доставки сообщений;
- лёгкая интеграция новых устройств, что делает протокол идеальным решением для Промышленного Интернета вещей (IIoT);
- для обеспечения безопасности в MQTT реализованы такие механизмы защиты

как: аутентификация клиентов по паре USERNAME и PASSWORD, подключение через SSL/TLS.

Протокол MQTT и безопасность информации X.509 сертификатов

При реализации механизма односторонней аутентификации сертификат с закрытым ключом находится у сервера [17,18]. Данные зашифровываются открытым ключом и могут быть расшифрованы только закрытым ключом сервера. Прежде чем установить соединение с сервером, клиент проверяет его сертификат.

Если сервер аутентифицирован, то его сообщение о сертификации должно обеспечить сертификационную цепочку, ведущую к доверенному центру сертификации. То есть, аутентифицированный сервер должен отправить клиенту сертификат, подписанный одним из центров сертификации, поэтому клиент также должен хранить список организаций CA, которым он доверяет.

Одностороння аутентификация, является менее безопасной, из-за на отсутствия механизма проверки клиентских сертификатов. Однако этот тип будут наиболее подходящим для систем, в частности, ввиду использования меньшей вычислительной мощности на смарт-устройства, обладающих низкой производительностью. Необходимо повышенное внимание, поскольку это может привести к более катастрофическим последствиям, чем нарушения упомянутые выше. Так неправильно настроенная политика авторизация клиентов на сервере злоумышленник сможет с лёгкостью перехватить управление микроконтроллером, при помощи отправки контрольных сообщений брокеру.

Построение собственного сегмента промышленного интернета вещей и обеспечение его безопасности

К моменту написания настоящей статьи поисковой запрос «MQTT» в поисковой системе по Интернету вещей Shodan выдавал порядка 150 000 результатов. Это число даёт примерное представление о количестве.

Чтобы понять, как обеспечивается защита на практике, в рамках представленной работы были произведены попытки отправки запросов на аутентификацию случайно выбранным с помощью символа «#». После анализа об используемом внутри системы оборудовании, его версии прошивки, MAC-адрес устройства (рис. 22). Все эти данные в теории могут помочь злоумышленнику сформировать вектор атаки на целевую систему.

Более серьёзную опасность представляет смарт-устройством (например, отправить сообщение открыть гаражную дверь командой «open» используя топик «home/garage_door»).

Выше изложенные сценарии атаки применимы в случае, когда сеть Интернета вещей обладает такой уязвимостью как слабая или недолжная конфигурация MQTT-брокера, что в условиях Промышленного Интернета вещей может привести к катастрофическим последствиям.

Далее будет рассмотрен пример создания и правильной конфигурации небольшого сегмента сети Промышленного Интернета вещей. Топология нашей сети с функционирующими внутри неё клиентами отражена на рис. 23.

Вся сеть будет завязана на взаимодействии клиентов с MQTT-брокером Mosquitto. Выбор брокера Mosquitto основывается на том, что он представляет собой брокер сообщений с открытым исходным кодом, который устанавливается локально и подходит для использования на всех устройствах, от одноплатных устройств с низким энергопотреблением до полноценных серверов. Настройка безопасности MQTT-брокера включает в себя следующие механизмы:

- аутентификация на основе логина и пароля (в этом случае необходимо вручную создать учетные данные клиентов на стороне брокера);
- авторизация клиентов на основе списка управления доступом (ACL);
- дополнительная односторонняя SSL аутентификация (данный механизм безопасности позволяет защитится от такой угрозы, как подмена сертификата сервера «человеком посередине»);

- TLS шифрование передаваемых данных для обеспечения конфиденциальности информации.

В качестве устройства, которое бы отправляло данные (основной издатель сети), будет выступать микроконтроллер китайского производителя Espressif Systems с интерфейсом Wi-Fi - ESP8266, на базе платы NodeMCU (рис. 24).

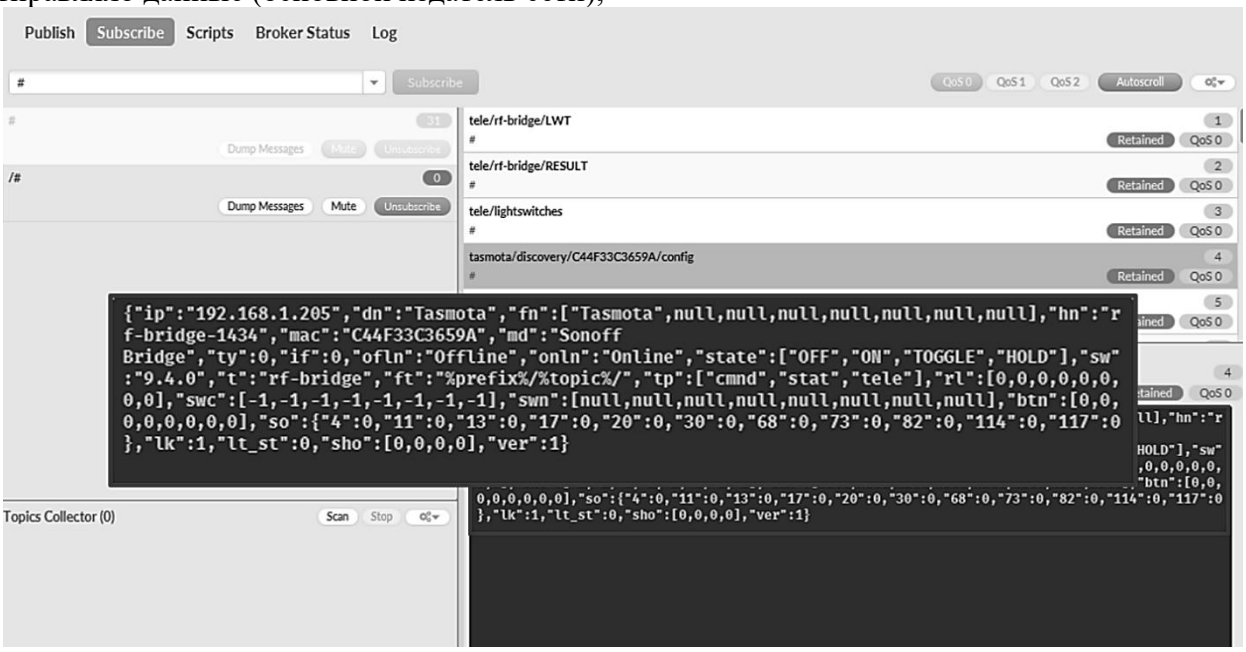

Рис. 22. Перехваченные данные об оборудовании и прошивке

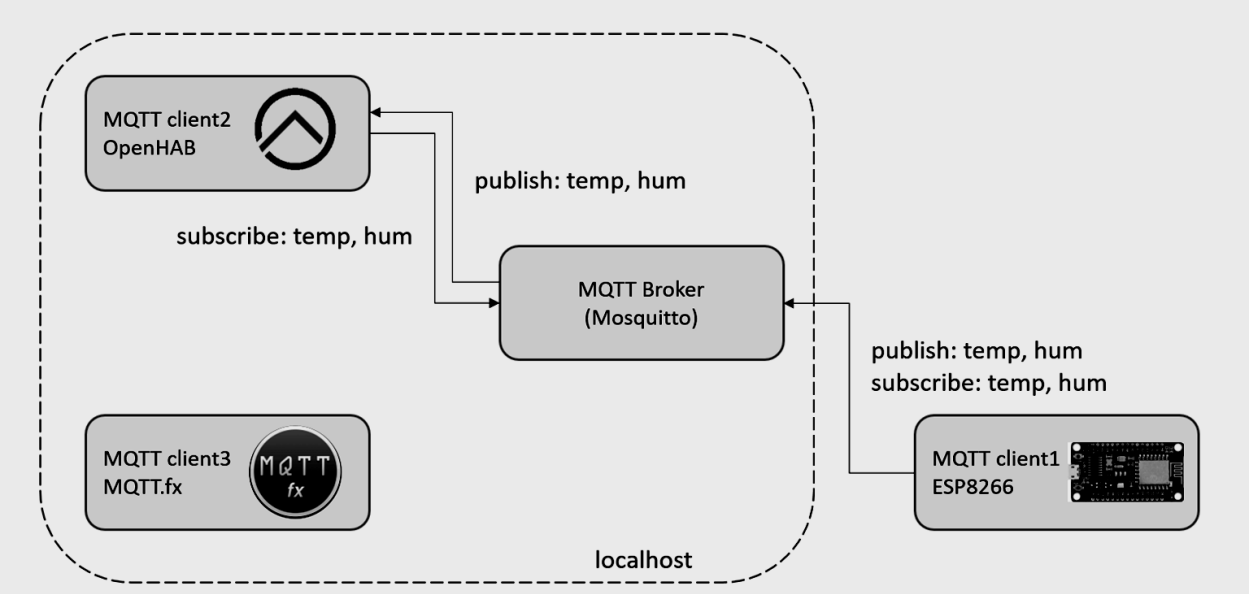

Рис. 23. Топология создаваемой сети IIOT

Для сбора данных используется цифровой датчик влажности и температуры DHT11, состоящий из термистора и емкостного датчика влажности. 2%), но зато недорог и прост в использовании. Собираемые датчиков измерения температуры и влажности микроконтроллер публикует при помощи соответствующих топиков. Схема подключения датчика к плате NodeMCU изображена на рис. 25.

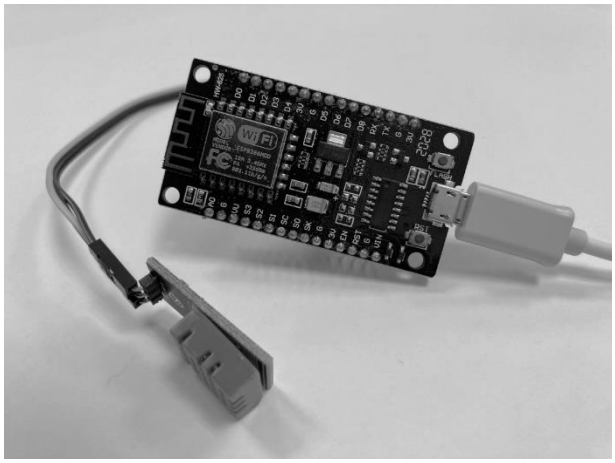

Рис. 24. NodeMCU ESP8266 с датчиком влажности и температуры DHT11

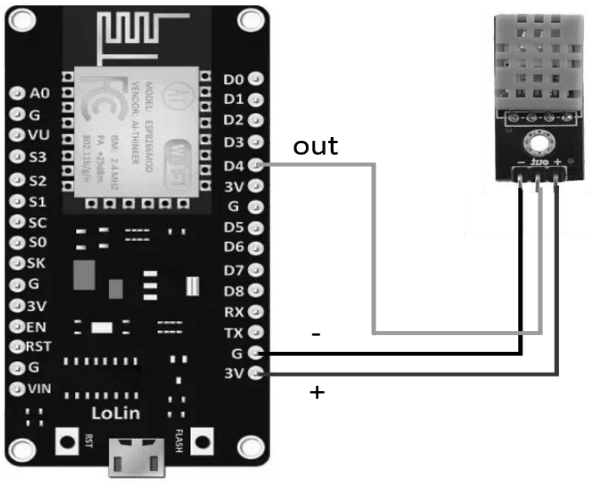

Рис. 25. Схема подключения датчика DHT11 к микроконтроллеру

Клиент сети OpenHAB подписывается на топики, отправляемые микроконтроллером, обрабатывает данные и визуализирует их. В теории можно было бы настроить определённую логику, при которой после анализа клиент OpenHAB сам бы отдавал команды другим смарт-устройствам посредством публикации сообщений. К примеру нашу сеть Интернета вещей можно было бы дополнить умным кондиционером, начинающим работать при получении контрольного MQTT-сообщения от клиента OpenHAB.

Построение и безопасная настойка конфигурации сегмента Промышленного Интернета вещей работы:

- сбор устройства для измерения температуры и влажности;
- настройка механизма аутентификации клиентов по логину/паролю и политики авторизации на основе списка управления доступом;
- написание скрипта для микроконтроллера ESP8266;
- настройка клиента OpenHAB на работу с брокером, создание устройств и привязка их к каналам;
- создание пары открытый/закрытый ключ и сертификатов X.509 сервера;
- настройка брокера Mosquitto, клиента OpenHAB и микроконтроллера ESP8266 на работу с сертификатами X.509 для односторонней аутентификации и TLS шифрования;
- контрольное тестирование защищённого соединения при помощи сетевого анализатора Wireshark.

Первоначальная настройка брокера предполагает собой добавление необходимых параметров в файл конфигурации брокера.

Для того, чтобы клиенты могли подключиться к нашему брокеру только по логину и паролю, необходимо создать конфигурационный файл, который будет содержать имена пользователей и зашифрованные пароли. Команда для создания файла с логином и паролем первого клиента выглядит следующим образом: mosquitto_passwd -c <passwordfile> <username>, где:

- –c – параметр, указывающий на создание нового файла, содержащим пару <username>/<password>;
- <passwordfile> – путь до конфигурационного файла passwd, где будут хранится логины и пароли;
- <username> – логин клиента.

Для добавления новых пользователей сети, параметр –c необходимо заменить на –b и дополнительно после указания логина клиента (<username>) добавить через пробел пароль клиента (<password>). Синтаксис в этом случае выглядит следующим образом: mosquitto_passwd –b <passwordfile> <username> <password>. Пример создания конфигурационного файла с логинами и зашифрованными паролями трёх клиентов показан на рис. 26.

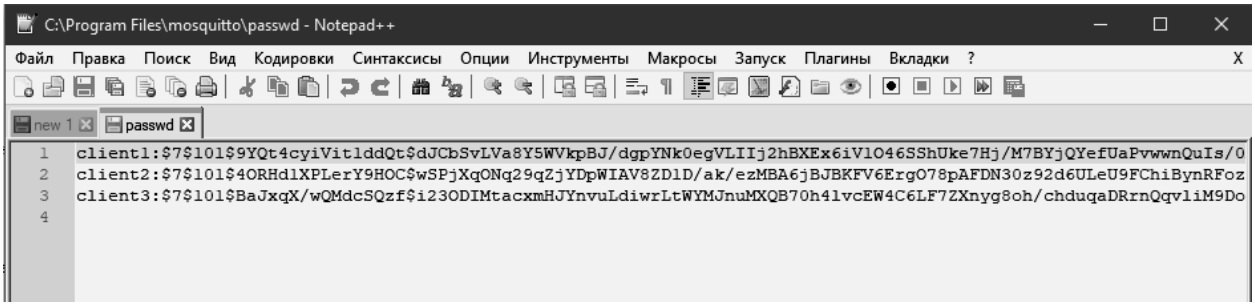

Рис. 26. Учётные данные созданных клиентов в конфигурационном фале «passwd»

После настройки механизма аутентификации клиентов, необходимо настроить политику авторизации, чтобы обозначить на какие топики клиенты могут подписываться, а какие топики публиковать. Ограничение тем выполняется в файле списка управления доступом, для этого в нём предусмотрено три раздела:

- общий раздел. Записи здесь начинаются с ключевого слова «topic» и применяются ко всем клиентам, за исключением случаев, когда клиент предоставил имя пользователя;
- специфичный для пользователя раздел. В этом разделе ключевым словом является «user». Раздел применяется только в случае если клиент предоставил имя пользователя. У каждого пользователя могут быть записи, определяющие его права доступа;
- раздел клиента или идентификатора пользователя. Каждый клиент предоставляет идентификатор клиента, поэтому записи в этом разделе будут применятся только к клиенту с этим идентификатором. Пример записи: pattern write $SYS/broker/%c/temp. Это означает, что клиент с идентификатором client1 может опубликовывать сообщения в топике $SYS/broker/client1/temp.

В нашем случае, права доступа трёх клиентов будут ограничиваться специфичным для пользователя разделом согласно рис. 27.

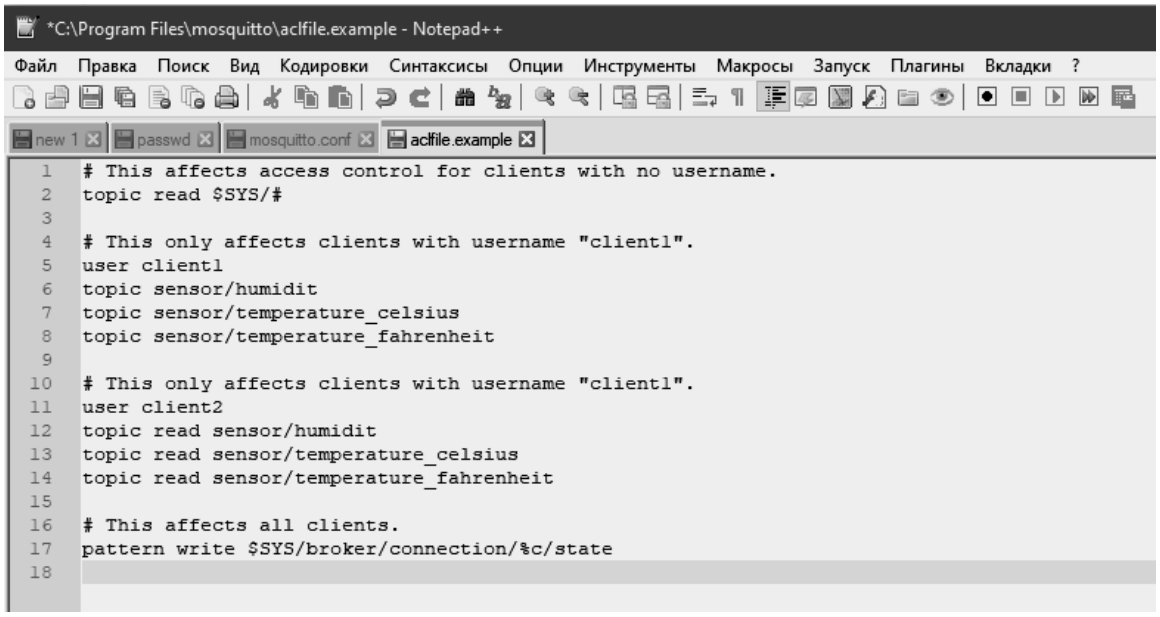

Рис. 27. Конфигурационный файл списка управления доступом

Теперь можно приступать к изменениям общего файла конфигурации брокера mosquitto.conf, в который необходимо добавить следующие параметры:

- listener <port> – указываем порт по которому будут подключаться клиенты и на котором будет находится служба брокера (в нашем случае это стандартный порт 1883);
- allow_anonymous false – параметр отключающий анонимное подключение к брокеру, если его значение false. Теперь для

подключения, подключение нужно будет предоставить пару логин/пароль;

- password_file <path to passwordfile> – для того, что бы брокер мог инициализировать клиентов, указываем ему путь до созданного файла конфигурации с учётными данными;
- acl_file <path to aclfile> – путь до файла со списками управления доступом (ACL).

Следующим шагом будет написание скрипта для микроконтроллера. Пример написанного скрипта, который инициализирует подключение к сети Wi-Fi, брокеру, замеряет температуру, влажность и публикует измерение с помощью соответствующих топиков,

На основе сравнения, в качестве был выбрал OpenHAB.

Сервер будет находится на одном хосте с брокером, согласно приведённой выше топологии сети, а подключение к нему будет производится через веб-браузер при помощи IP адреса и порта (в нашем случае порт 8080). В ходе работы с OpenHAB, был создан интерфейс, предназначенный для мониторинга показателей (температуры и влажности) серверной. Сам процесс создания и настройки интерфейса мониторинга серверной весьма сложный на первый взгляд, ввиду запутанности и отсутствия русского языка, предполагающий собой создание «айтемов» (рис. 28), «вещей», каналов и страниц (рис. 29).

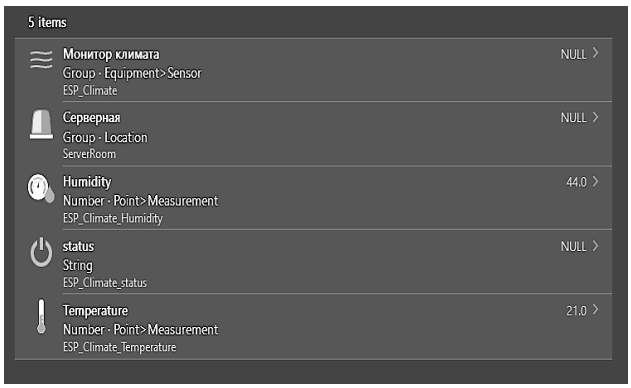

Рис. 28. Созданные «айтемы»

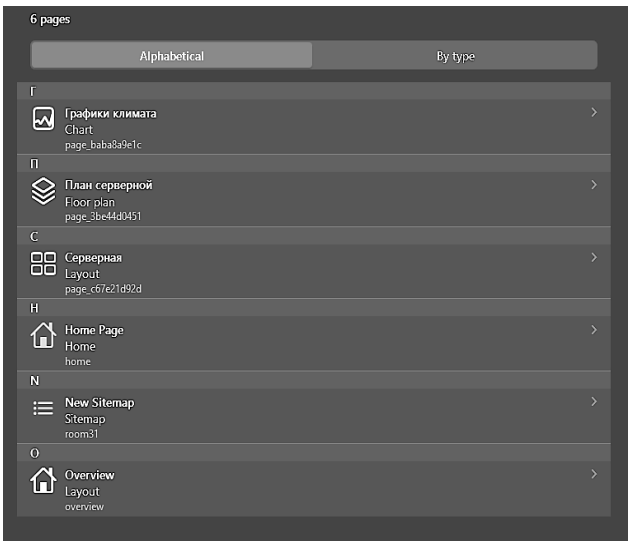

Рис. 29. Созданные вкладки

Заключительным этапом создания и первоначальной настройки сети будет загрузка скрипта в микроконтроллер и его выполнение. Для того, чтобы убедится в правильности работы микроконтроллера, необходимо зайти в монитор порта, который представляет собой утилиту, встроенную в среду программирования Arduino IDE и служит для связи компьютера с контроллером (рис. 30).

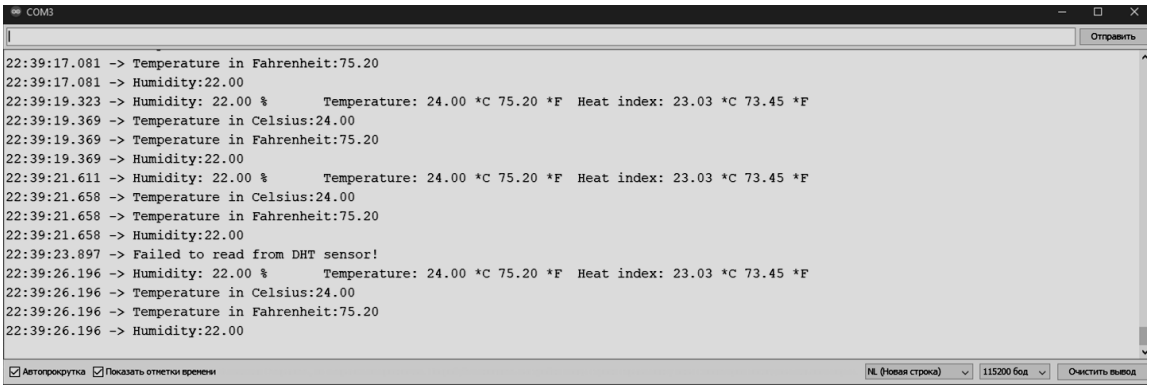

Рис. 30. Монитор последовательного порта Arduino IDE

Тем временем клиент OpenHAB уже успешно принимает данные от микроконтроллера и в интерфейсе мониторинга климата серверной можно наблюдать визуализацию первых измерений во вкладках «Начальная страница» (рис. 31), «План серверной» (рис. 32) и «Графики климата» (рис. 33).

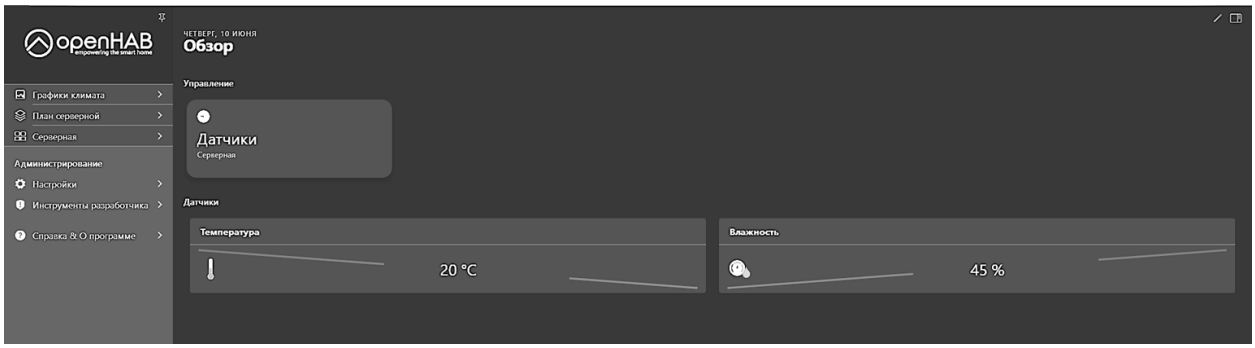

Рис. 31. Начальная страница OpenHAB

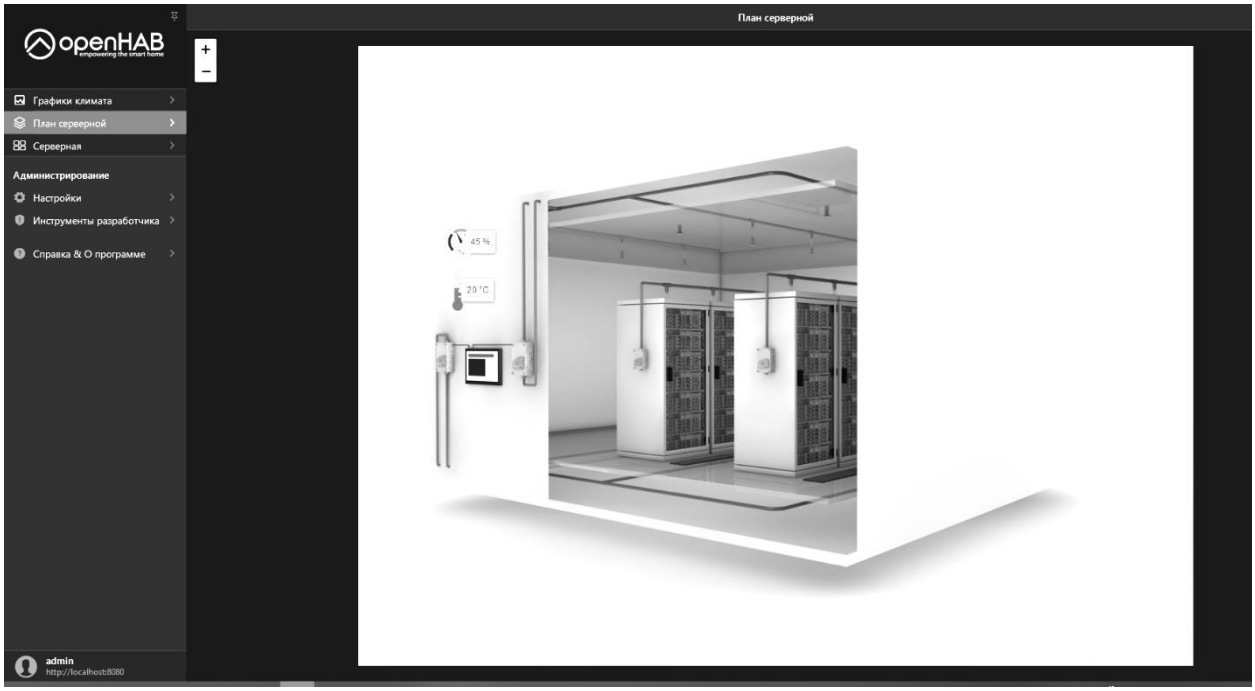

Рис. 32. Вкладка «План серверной» с отображением показателей

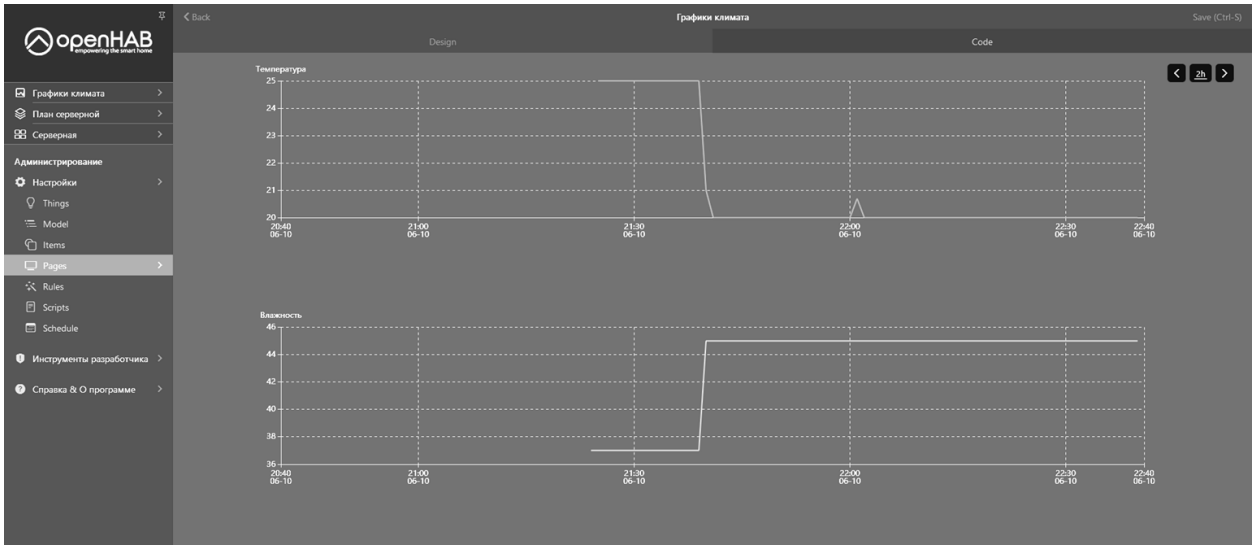

Рис. 33. Вкладка «Графики климата»

Воспользуемся сетевым анализатором трафика Wireshark, для того что бы захватить и проанализировать MQTT пакеты. Как ранее говорилось, MQTT работает поверх TCP, обеспечивая необходимый уровень надёжности, однако данные, как видно из рис. 34, публикуются в открытом виде.

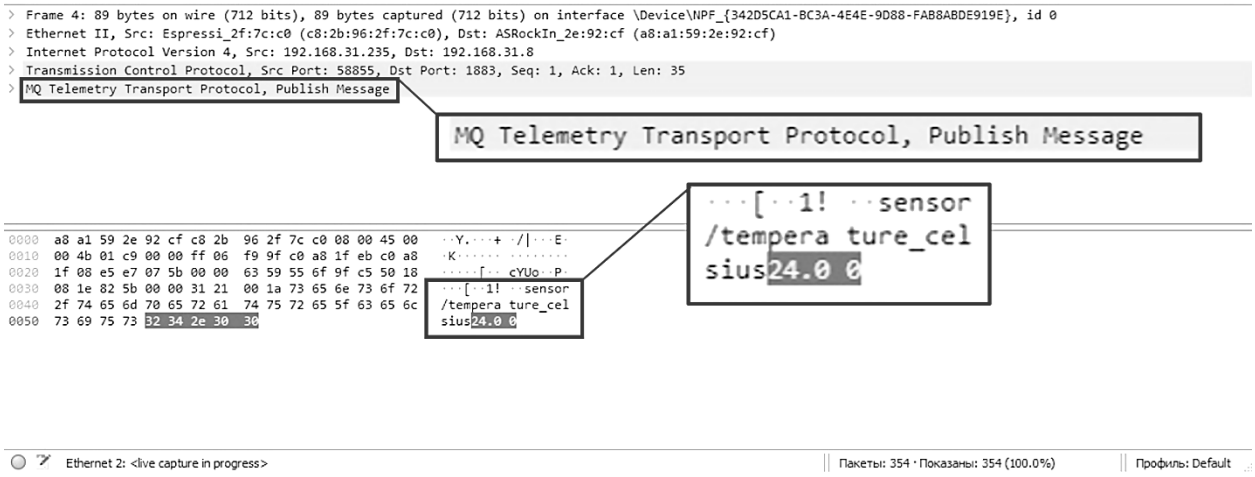

Рис. 34. Перехваченный пакет MQTT в открытом виде

При помощи добавления механизма аутентификации и настройки политики авторизации в сети, получилось частично обеспечить конфиденциальность и доступность данных, однако сеть всё равно остаётся уязвимой для атаки «человек по середине», при которой нарушается не только конфиденциальность данных, но и их целостность. Обеспечить защищённое соединение между брокером и клиентами можно путём применения SSL (Secure Sockets Layer) и TLS (Transport Layer Security).

TLS и его предшественник SSL - это криптографические протоколы, предназначенные для обеспечения безопасности связи в компьютерных сетях.

По своей сути использование SSL/TLS обеспечивает шифрование данных, целостность данных и аутентификацию. У нас же, нет нужды использовать TLS/SSL в качестве аутентификации клиентов, так как у нас уже настроена аутентификация по логину и паролю. Вместо этого будет использоваться односторонняя аутентификация для проверки сертификата сервера и установки зашифрованного соединения. Участок сети MQTT-брокер – клиент OpenHAB будет защищён от атаки типа подмена сертификата сервера «человеком посередине», так как клиент OpenHAB при первом подключении к брокеру, вычисляет и сохраняет хэши сертификата и открытого ключа сервера, для сверки в последующих соединениях.

Для создания собственного центра сертификации (далее - CA), ключей и сертификатов будет использоваться криптографическая библиотека с открытым исходным кодом OpenSSL, широко известна из-за расширения SSL/TLS.

Для установки зашифрованного соединения сервер должен обладать:

- сертификатом центра сертификации;
- сертификатом сервера брокера Mosquitto, подписанный сертификатом центра сертификации;
- секретным ключом сервера для расшифровки.

Тогда как для клиента, требованием будет лишь наличие сертификата центра сертификации, для проверки сертификата брокера Mosquitto.

Для того, чтобы установить защищённое соединение с использованием TLSv1.2 необходимо выполнить следующие шаги:

- создание пары ключей CA;
- пары ключей брокера, не защищённых паролем (openssl genrsa -out server.key 2048);
- создание запроса сертификата брокера, используя ключ брокера (openssl req -new -out server.csr -key server.key);
- подписание запроса сертификата брокера, используя сертификат CA;(
- теперь в наличии есть файл ключа CA, сертификат CA, файл ключа брокера, сертификат брокера;

• перемещаем сертификат CA, файл ключа брокера и сертификат брокера в созданный каталог внутри брокера (каталог /cert, рис. 35);

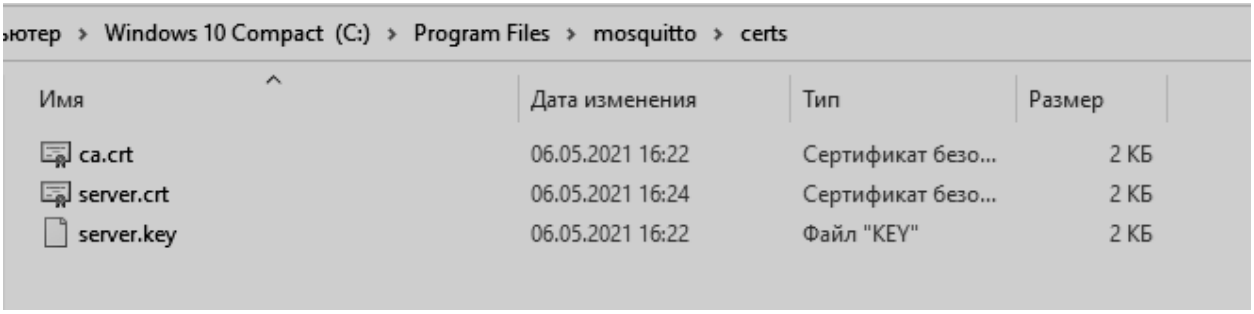

Рис. 35. Каталог «certs» Mosquitto

• копирование файла сертификата CA клиентам для проверки сертификата сервера брокера;

• редактирование конфигурационного файла mosquitto.conf для работы с TLSv1.2 (рис. 36). Для этого в нём необходимо добавить следующие параметры:

1) port 8883 – указываем порт для защищённого соединения вместо порта 1883;
2) cafile <path to ca.crt> - путь до файла сертификата CA;
3) certfile <path to server.crt> - путь до файла сертификата брокера Mosquitto;
4) keyfile <path to server.key> - путь до секретного ключа сервера брокера;
5) tls_version tlsv1.2 – указываем используемую версию TLS;

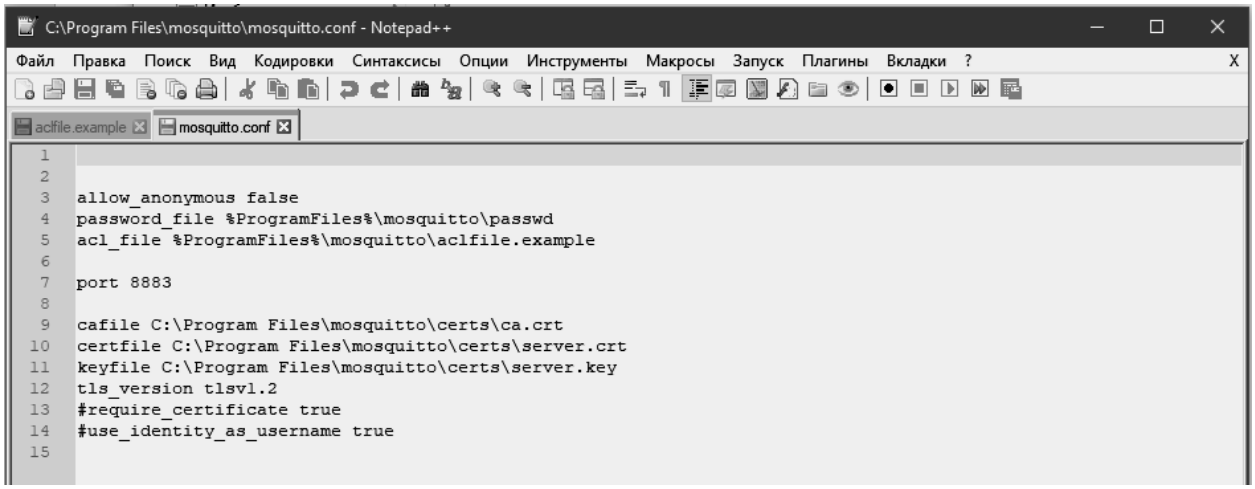

Рис. 36. Главный конфигурационный файл брокера

• изменение клиентского сценария OpenHAB, чтобы использовать TLS и сертификат CA, для этого необходимо установить параметр «Secure Connection» а также изменить порт для подключения к брокеру. После первого подключения к MQTT-брокеру, клиент OpenHAB запоминает хэши сертификата сервера и его открыто ключа для проверки при следующих подключениях к брокеру (рис. 37);

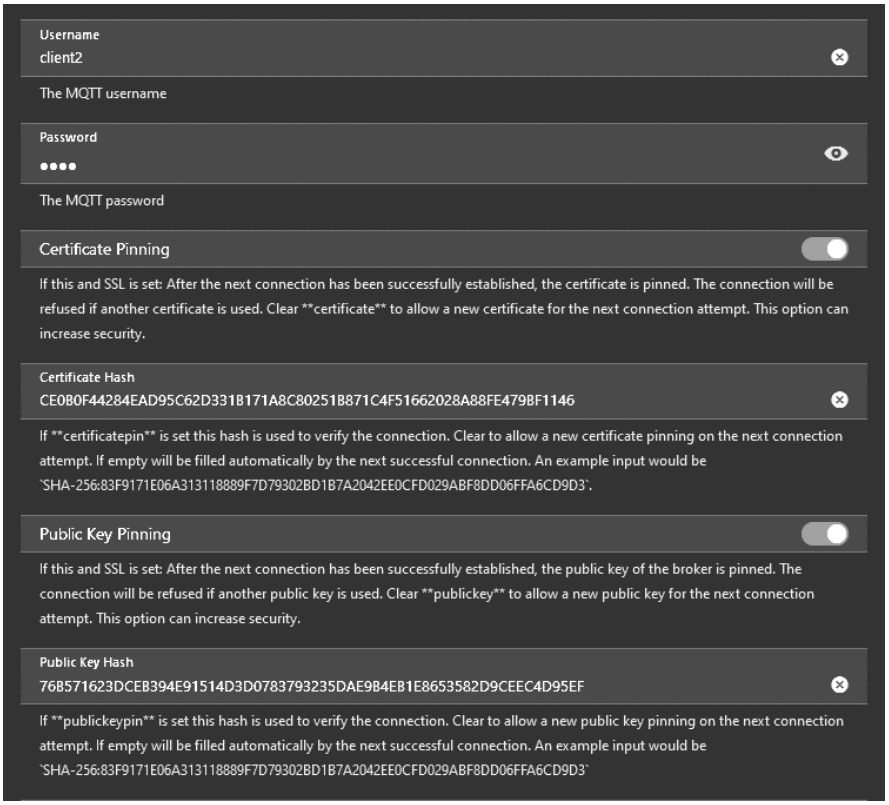

Рис. 37. Настройка механизма SSL – pinning

- добавление в скрипт микроконтроллера необходимых библиотек для работы с TLS, создание макроса для проверки сертификата сервера в виде статической константы. Пример написанного скрипта для защищённого подключения к брокеру, измерения температуры и влажности, представлен.

После того как все настроено и работает, возвращаемся к сетевому анализатору пакетов Wireshark, который показываем нам захваченные пакеты. Теперь в стеке протоколов дополнительно можно наблюдать TLSv1.2 и измерения в зашифрованном виде (рис. 38).

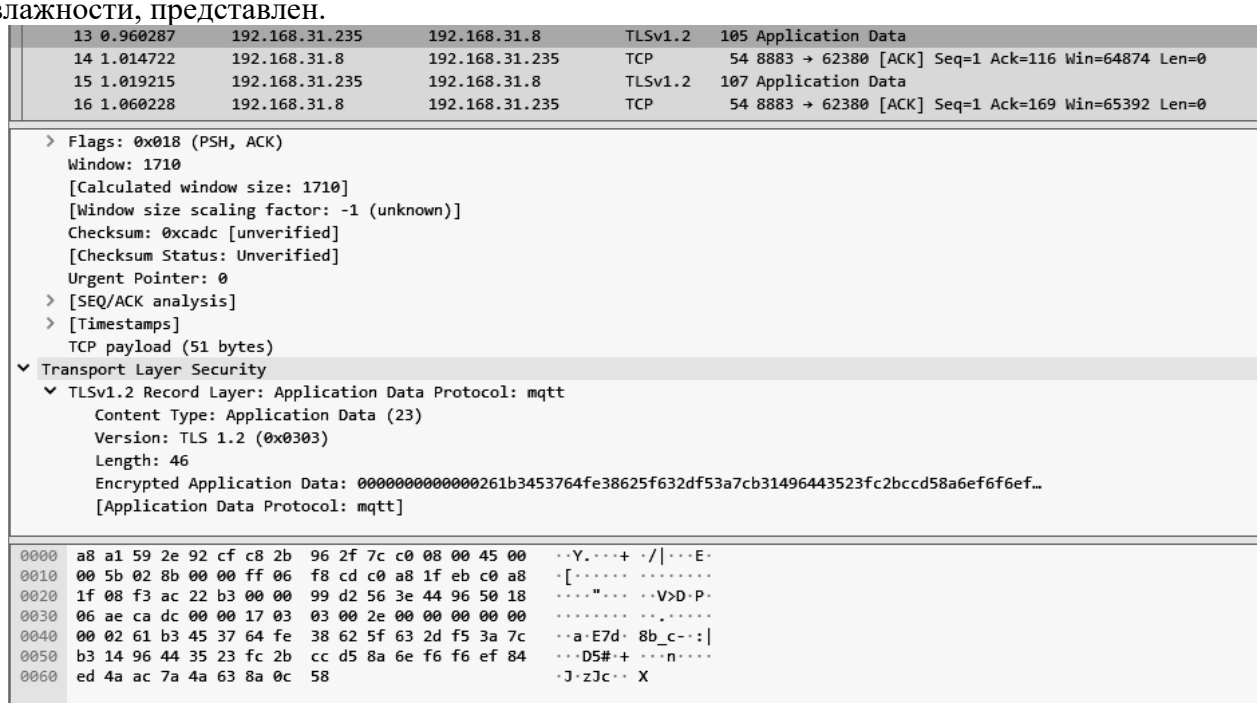

Рис. 38. Перехваченный пакет MQTT с данными в зашифрованном виде

- **Заключение**

В представленной статье показаны результаты проведенных исследований по обеспечению информационной безопасности киберфизических систем и интернет вещей, как их частный случай. Как видно из представленных материалов, проблема обеспечения информационной безопасности киберфизических систем является серьезной, включающей в себя целую совокупность сложнейших задач, и требует комплексного подхода к ее решению.

Список литературы

1. Черняк Л. Киберфизические системы. Cyber-Physical System, CPS // TADVISER. Государство. Бизнес. IT. URL: http://www.tadviser.ru/index.php / Статья:Киберфизические системы (Cyber-Physical System, CPS) (дата обращения: 02.05.2021).

2. Башелханов И.В. Обеспечение информационной безопасности в киберфизических системах в условиях Индустрии 4.0 / И.В. Башелханов, Н.И. Демкина, С.М. Володин, А.В. Рой, А.В. Олескин, В.Н. Решетников // Информационная безопасность: вчера, сегодня, завтра: сборник статей по материалам Междунар. науч.-практ. конф. М: Российский государственный гуманитарный университет, 2019. С. 153-160.

3. Указ Президента РФ от 9 мая 2017 г. № 203. О Стратегии развития информационного общества в Российской Федерации на 2017 - 2030 годы.

4. Павленко Е.Ю. Обеспечение информационной безопасности киберфизических систем на основе принципа гомеостаза: дисс. на соискание ученой степени канд. техн. наук: 05.13.19 / Павленко Евгений Юрьевич. СПБ., 2018. 183с.

5. Атаки на киберфизические системы. // tadviser. URL: https://www.tadviser.ru (дата обращения: 09.06.2021).

6. toshiba_ru. Кибер-физические системы в современном мире. [Электронный ресурс] // Хабр. URL: https://habr.com/ru/company/toshibarus/blog/438262/ (дата обращения: 03.05.2021).

7. CDTOCenter. Умная нация, или чем интересен опыт цифровой трансформации Сингапура? // Хабр. URL: https://habr.com/ru/company/cdtocenter/blog/530154/ (дата обращения: 03.05.2021).

8. Коми подключается к проекту «Умный город». // Правительство Республики Коми. URL: https://gov.rkomi.ru/node/1537 (дата обращения: 03.05.2021).

9. di1453. Промышленный интернет вещей: рассказываем об успешных кейсах. // Хабр. URL: https://habr.com/ru/company/kauri_iot/blog/471588/ (дата обращения: 09.06.2021).

10. Федотенков С. 10 самых впечатляющих кибератак в истории // 3Dnews: URL: https://3dnews.ru/1009634/10-samih-vpechatlyayushchih-kiberatak-v-istorii (дата обращения: 04.05.2021).

11. Как начать работу с iRidium. // iRidium mobile. URL: http://wiki2.iridiummobile.ru/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0 (дата обращения: 02.06.2021).

12. admin. Iridium Mobile – система управления «умным» домом. // smart home. URL: https://umniedoma.ru/iridium-mobile-sistema-upravleniya-umnym-domom/ (дата обращения: 02.06.2021).

13. morfeusys. OpenHAB – стань программистом собственного жилища. // Хабр. URL: https://habr.com/ru/post/232969/ (дата обращения: 07.05.2021).

14. CooperMaster. Шаблоны взаимодействия для интернета вещей. // Хабр. URL: https://habr.com/ru/company/intel/blog/397241/ (дата обращения 27.05.2021).

15. Гойхман В., Савельева А. Аналитический обзор протоколов Интернета вещей. // Технологии и средства связи. URL: http://lib.tssonline.ru/articles2/reviews/analiticheskiy-obzor-protokolov-interneta-veschey (дата обращения: 10.05.2021).

16. Что такое MQTT и для чего он нужен в IIoT? Описание протокола MQTT. // ipc2u. URL: https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/ (дата обращения 22.05.2021).

17. Я по ИБ и немного по ИТ. Протоколы Интернета вещей (какая-то небольшая часть из них). // Яндекс Дзен. URL:

https://zen.yandex.ru/media/id/5ba14cc142ef5c00af0d97f6/protokoly-interneta-vescei-kakaiato-nebolshaia-chast-iz-nih-5f2af7457f7edb5a706a2d0d (дата обращения: 15.05.2021).

18. Криницин В.В. Протокол MQTT как основа шаблона «издатель – подписчик» в Интернете вещей / В.В. Криницин // Информационно-методический журнал INSIDE Защита информации. 2020. № 5. С.72-76.

Московский государственный лингвистический университет
Moscow State Linguistic University

Сыктывкарский государственный университет им. Питирима Сорокина
Syktyvkar State University after the Pitirim Sorokin

Поступила в редакцию 04.08.2022

Информация об авторах

Филяк Петр Юрьевич – канд. техн. наук, директор Института информационных наук Московского государственного лингвистического университета, e-mail: p,filiak@linguanet.ru

Тырин Илья Алексеевич – магистрант направления «Математика и компьютерные науки», бакалавр направления подготовки «Информационная безопасность», Сыктывкарский государственный университет, e-mail: delkan665@rambler.ru

Пажинцев Денис Алексеевич – бакалавр, направление подготовки «Информационная безопасность», Сыктывкарский государственный университет, e-mail: denispazh@gmail.com

INFORMATION SECURITY OF CYBER-PHYSICAL SYSTEMS

P.Y. Filyak, I.A. Tyrin, D.A. Pazhintsev

The active introduction of information technologies, such as cloud storage or the Internet, e-commerce, in the modern world creates extensive prerequisites for an increase in the number of "smart" devices, and, consequently, the formation of various cyber-physical systems (CPS). Such systems are already actively used in various spheres of life: medicine, industry, military structures, CII, etc. Due to the fact that the concept consists of many technological trends, one of the key areas was chosen for consideration – in particular, cyber-physical systems. The relevance of choosing this segment and the research topic in general is justified by such factors as the rapid growth in the number of "smart" devices, as a result of which the number of impacts and attacks on devices and systems as a whole increases. Therefore, CPS information security is one of the most popular areas of study and research today.

Keywords: cyber-physical systems, CPS, information security, interface, protocols, data transmission channels, secure information systems, interception control channels, intruders.

Submitted 04.08.2022

Information about the authors

Petr Yu. Filyak – Cand. Sc. (Technical), Director of the Institute information Sciences, Moscow State Linguistic University, e-mail: p.filiak@linguanet.ru

Iliya A. Tyrin – Undergraduate of Mathematics and computer science, Bachelor of Information Security, Syktyvkar State University, e-mail: delkan665@rambler.ru

Denis A. Pazhintsev – Bachelor of Information Security, Syktyvkar State University, e-mail: denispazh@gmail.com