Geekbrains

Создание стенда для проверки – контроля работы контроллерной платы управления лифтом на примере платы HD X ONE.

Программа:

ИОТ

Фролов А.Ю.

Санкт-Петербург

2024

Содержание

Оглавление

Введение

Тема проекта:

Разработка базового универсального инструментария для компьютерного тестирования плат управления лифтом на примере платы от производителя HEDEFSAN (STEIMBERG) платы HD ONE. Первый этап: разработка контроля – считывание основных рабочих параметров, т.е. рабочих состояний выходов и входов платы управления во время имитации работы лифта. В дальнейшем, вне рамок дипломного проекта, – создание стенда для симуляции рабочих состояний оборудования для диагностики правильности работы платы управления лифта.

Цель:

Изучить особенности работы платы для создания универсального тестового (контрольного) оборудования плат такого же типа. Написать код для обработки рабочего процесса платы. Проработать вариант (по временному интервалу выбранных событий) обработки и сохранения событий в базу данных.

Какую проблему решает:

Иногда возникают проблемы работы плат управления, которые явно не видны, либо был произведен ремонт платы, и необходимо убедиться в работоспособности (корректной работы) платы, при этом, требуется тестирование работы в определенный промежуток времени отработка определенных ситуаций работы оборудования.

Задачи:

1. Изучить литературу, касающуюся темы проекта.
2. Рассмотреть основные виды оборудования для использования в проекте, подобрать варианты исполнения.

3. Ознакомиться с основными принципами составления различной документации для описания работы и проведения тестов,- например: мануал, чек-лист, тест-кейс, тест-план, баг-репорт.

4. Составить план для тестирования плат и выведения алгоритма теста, определить контролируемые тестовые параметы, учесть при этом аспекты для расширения и увеличения тестовых возможностей.

5. Написать код для теста работы платы Hedefsan (STEIMBERG) HD X ONE.

6. Рассмотреть возможность взаимодействия с сервисом хранения данных на протоколе MQTT.

7. Разработать методику анализа контрольных данных с выводом результата в базу данных типа Influx DB или аналогичную. Рассмотреть возможность работы с Node-Red.

8. Разработать предложения по улучшению и расширению объема тестирования параметров лифтовых плат для приложения. Проработать основы для разработки web интерфейса для дистанционного контроля.

Инструменты:

среды разработки MS Visual Code и Geany, операционная система LINUX, клиент Eclipse paho MQTT, база данных INFLUX DB, графическое отображение Grafana, сервис Node-red, микрокомпьютер типа Raspberry, платы расширения и гальванической развязки для управления и контроля.

Теоретическая и практическая главы

В моем повседневном внимании существует ряд проблем, которые необходимо решать. В частности – проверка корректности работы лифтового оборудования, настройка режимов работы лифта в различных условиях и при этом необходимо отследить реакцию систем управления лифтом на разные входные сигналы, убедиться в правильной работе системы. Лифтовое оборудование на сегодня очень разное, много производителей и каждый год постоянно совершенствуется и модернизируется. Но есть ряд стандартных наборов команд и приемов работы любой платы управления, есть определенный перечень требований, к работе лифтового оборудования заложенный в Технический Регламент.

В основе работы практически любой платы управления лифтом лежит определенный алгоритм обработки событий и произведение определенных действий с органами управления для осуществления перемещения кабины по шахте. Причем, есть много режимов работы лифта – пожарный режим, перевозки пожарных подразделений, режим сейсмо-опасности, погрузка, приоритетные вызовы, ограничение доступа пользования лифтом, тестовое движение, организованные и приоритетные зоны парковки, движение в связке с группой лифтов и многое другое.

Плата должна получать информацию с датчиков таких как:

- энкодер – для получения значения о местоположении кабины позволяет более точно выдавать команды управления и получать более точное позиционирование на остановках (чаще всего - есть, но существует возможность использования набора датчиков позиционирования);

- датчики нахождения кабины в зоне верхнего и нижнего уровней;

- датчики точной остановки (от 1 до 5), часто используют 2-а датчика;

- датчики контроля положения двери кабины (открыты/закрыты);

- датчики загрузки кабины;

- датчики от устройств разного назначения (о готовности работы с частотным преобразователем, о пожаре, землетрясении, систем доступа и т.п.);

- датчик температуры (от лебедки и может быть также от др. оборудования);

- датчики ключа (для перевозки пожарных подразделений, приоритетных режимов);

- датчик наличия препятствий в зоне дверей (фотозавеса);

- датчик (2 микровыключателя) срабатывания тормоза лебедки.

Помимо этого, контролируется наличие отсутствия разрыва в цепи безопасности и нормальное состояние работы в цепи безопасности дверей шахты и кабины (при команде открыть дверь – цепь размыкается, при команде закрыть двери – цепь замыкается). Должен осуществляться контроль одновременного открытия дверей шахты и кабины на этаже остановки. Также плата должна получать обратную связь о работе управляющих устройств – контакторы (силовые пускатели).

Плата управления, так же взаимодействует с периферийными платами на кабине, в приказной панели (для принятия приказов, управления открытием и закрытием дверей на остановке и др.) и в приямке (при наличии функции управления из приямка) и аппаратами вызова на остановках. При работе в групповом режиме – взаимодействовать с другими лифтами. Возможны варианты работы основной платы с компьютером для сохранения параметров или управления.

В данном проекте будет использован упрощенный набор функций и операций – т.к. прежде всего, необходимо осуществить контроль за основными действиями платы – управление выбором скорости движения, прохождение команды на пуск основного пускателя с контролем обратной связи его работы, срабатывание датчика точной остановки в движении и датчиков верхнего и нижнего уровня. Чаще, плата управления взаимодействует с частотным преобразователем для осуществления плавного пуска и остановки кабины этот случай наиболее актуален и будет рассмотрен в проекте. Для варианта без частотного преобразователя потребуется формировать сигналы: скорость 1, скорость 2, направление вверх/вниз. Для работы с частотным преобразователем (далее ЧП) используют команды для выбора скорости (3 бита), выбор направления вверх/вниз (2 бита) и сигнал для разрешения движения (1 бит).

Рассмотрим достаточно простую, но функциональную плату турецко-российского производителя STEIMBERG (см. фото 1).

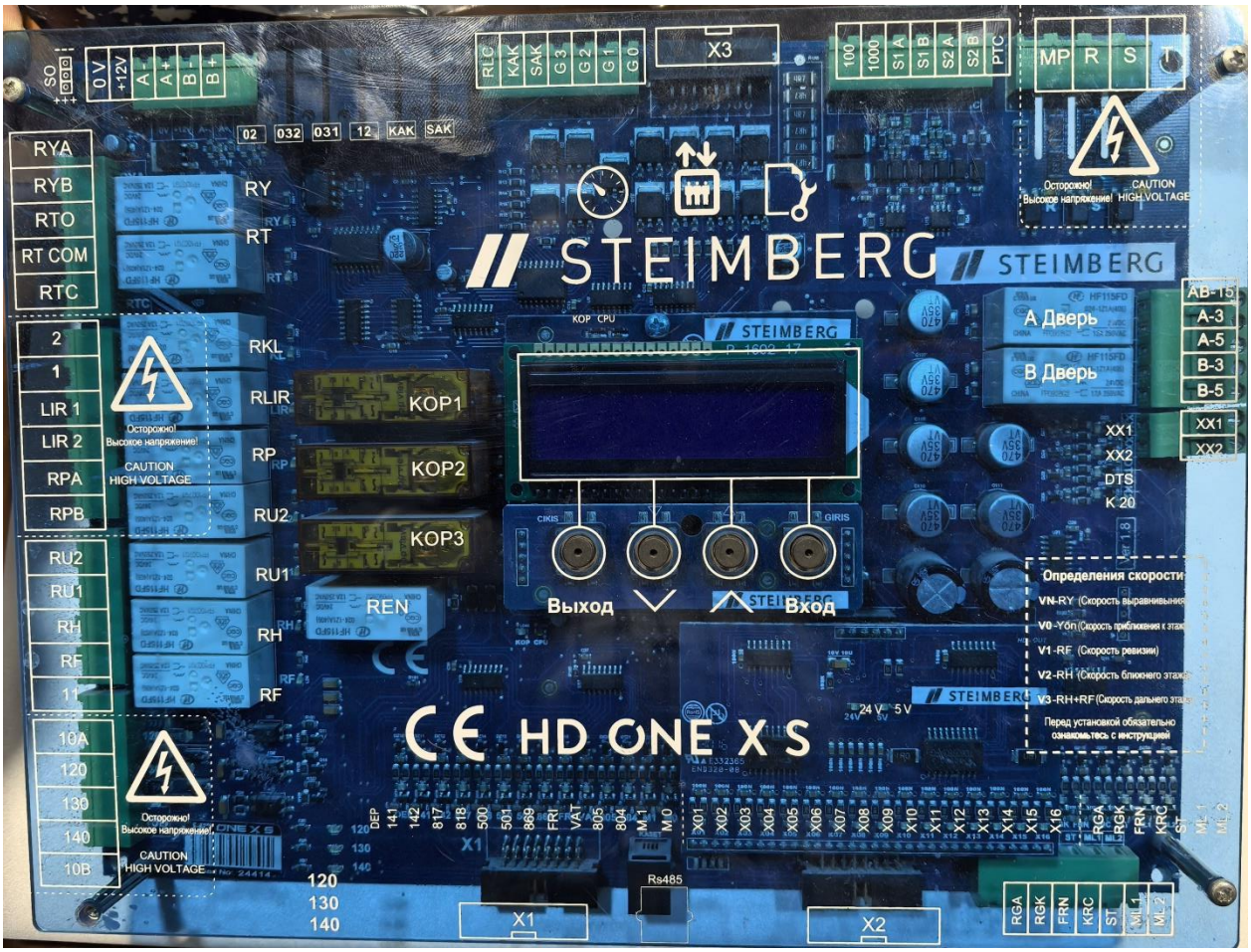


Фото 1 – внешний вид платы управления.

Таким образом, для организации контроля работы выходов с данной платы, потребуется использовать следующие контрольные входа:

5 бит для работы с ЧП скорость и направление (RY, RH, RF, RU1, RU2);

1 бит – контроль пуска основного пускателя (RGK);

2 бит – состояние датчиков точной остановки (ML1 ML2);

1 бит режим ревизии (869);

2 бит направление движения в ревизии (500, 501);

1 бит контроля срабатывания тормоза лебедки (FRN);

1 бит контроля перегрева лебедки (PTC);

1 бит контроля команды управления открытием / закрытием дверей кабины (A3/A5);

2 бит положения верха / низа (818 / 817).

Дополнительно:

2 бит перегрузка/полная загрузка (OL, FL);

2 бит (FRI, VAT).

Итого: **16 входов для контроля** (с учетом дисплея и дополнительных функций – 24 входа).

Из комбинации состояний этих сигналов получается описание состояния текущего рабочего режима платы.

*Не учитываем возможности платы по получению сигнала от энкодера (считываем код для дисплея), наличие взаимодействия с оборудованием по шине RS485 – в виду отсутствия информации от производителя о формате и расшифровке передаваемых данных.

Плата выдает информацию о текущем этаже с помощью цифровых выходов для этажного дисплея, код может быть представлен в формате Грей кода или бинарного значения в 4 битах, в таком случае потребуется еще 4 бита для считывания этой информации.

Если рассматривать вариант дистанционного запуска - управления на проверочном стенде, то потребуются выходы для управления и симуляции движения. В минимальном рассмотрении потребуется:

2 бит для формирования импульсов от виртуального энкодера;

2 бит – состояние датчиков точной остановки (ML1 ML2);

1 бит режим ревизии (869);

2 бит направление движения в ревизии (500, 501);

2 бит положения верха / низа (817 / 818);

для симуляции режимов и имитации аварийного состояния:

1 бит контроля срабатывания тормоза лебедки (FRN);

1 бит контроля перегрева лебедки (PTC);

1 бит контроля пускателя (KRC);

1 бит пожарный режим (FRI);

1 бит – контроль срабатывания основного пускателя (обратная связь KRC);

2 бит для управления включением цепи безопасности (120, 140+130);

4 бит для команды вызова или приказа (может быть использован дешифратор для увеличения емкости выбора)

Итого: **16 выходов** + 4 для организации команд вызова и +4 для организации команд приказа (можно использовать минимум по 2 бита для вызова/приказа).

Рассмотрим варианты реализации тестового оборудования.

1. Платы на базе микропроцессора ESP 8266 + платы расширения IC2
2. Плата на базе микропроцессора семейства STM32
3. Микрокомпьютер типа Raspberry и платы расширения IC2
4. Что-то еще…

Вариант 1 – достаточно удобные для работы платы с возможность на каждой плате организовать свой сервер, но, если требуется организовать взаимодействие с разными

тестируемыми платами, потребуется оперативно менять программный код в плате и в веб-сервере, что не очень удобно.

Вариант 2 - позволяет не использовать платы расширения для увеличения портов входа, но опять же нет оперативной настройки под оборудование, потребуется перепрограммировать плату под каждый конкретный случай.

Вариант 3 – на мой взгляд, наиболее оптимальное решение, по - скольку позволяет расширять, при необходимости, возможности системы для тестирования, причем с помощью приложения клиента-обработчика можно настраивать систему для работы с конкретной тестируемой платой, выбрать для нее определенные, необходимые установки и настроить отображаемую и передаваемую информацию о плате, можно реализовать рабочее место на тестовом стенде для обработки событий и до-настройки под определенные тесты. И более удобно в обслуживании,- можно формировать набор файлов настроек для каждого тестируемого образца. И можно организовать любой вариант сохранения результатов тестирования как локальный, так и сетевой.

В обычном случае, у Raspberry PI 4 используется разъем GPIO на 40 контактов (рис. 1). Для тестирования использовался Orange PI 2 Zero с 26 контактами (рис. 2), однако выходов/входов, в любом случае,- недостаточно для реализации всех функций, поэтому имеет смысл применить устройство расширения, работающее по стандарту I2C, что позволит подключить до 63 устройств на линию. На каждом устройстве, например, на базе чипа PCF8574T можно использовать по 8 входов/выходов (рис. 3).

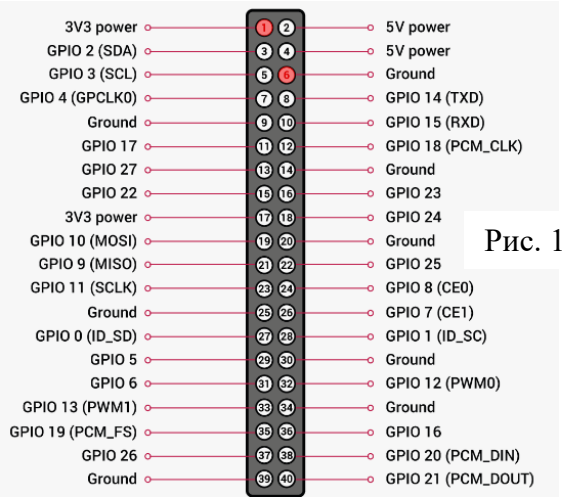


Рис. 1

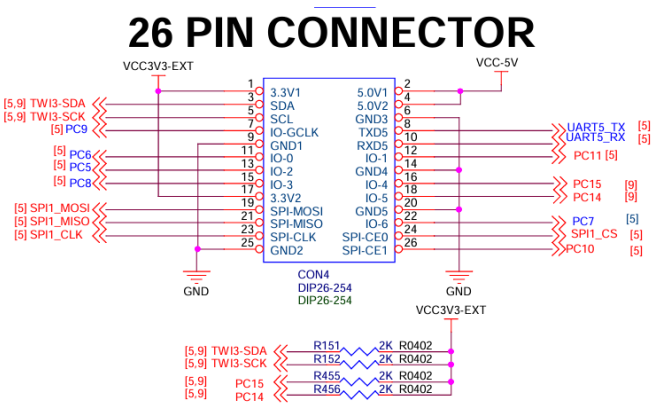
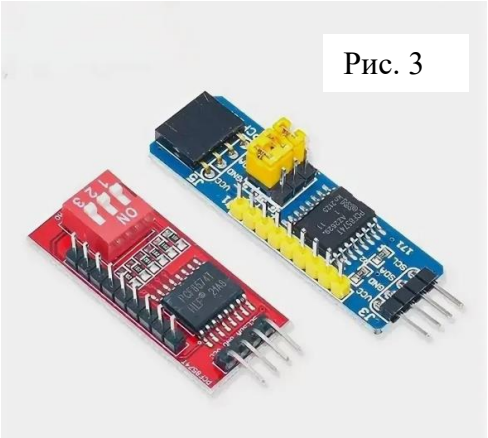


Рис.2

Рис. 3



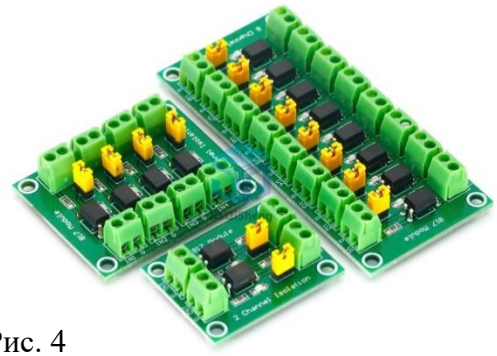


Рис. 4

Для гальванической развязки (в Raspberry 3.3В, а в системе управления используется 24В) необходимо использовать преобразователь уровней или делитель напряжения. В качестве такого преобразователя используем плату с оптронами (рис. 4).

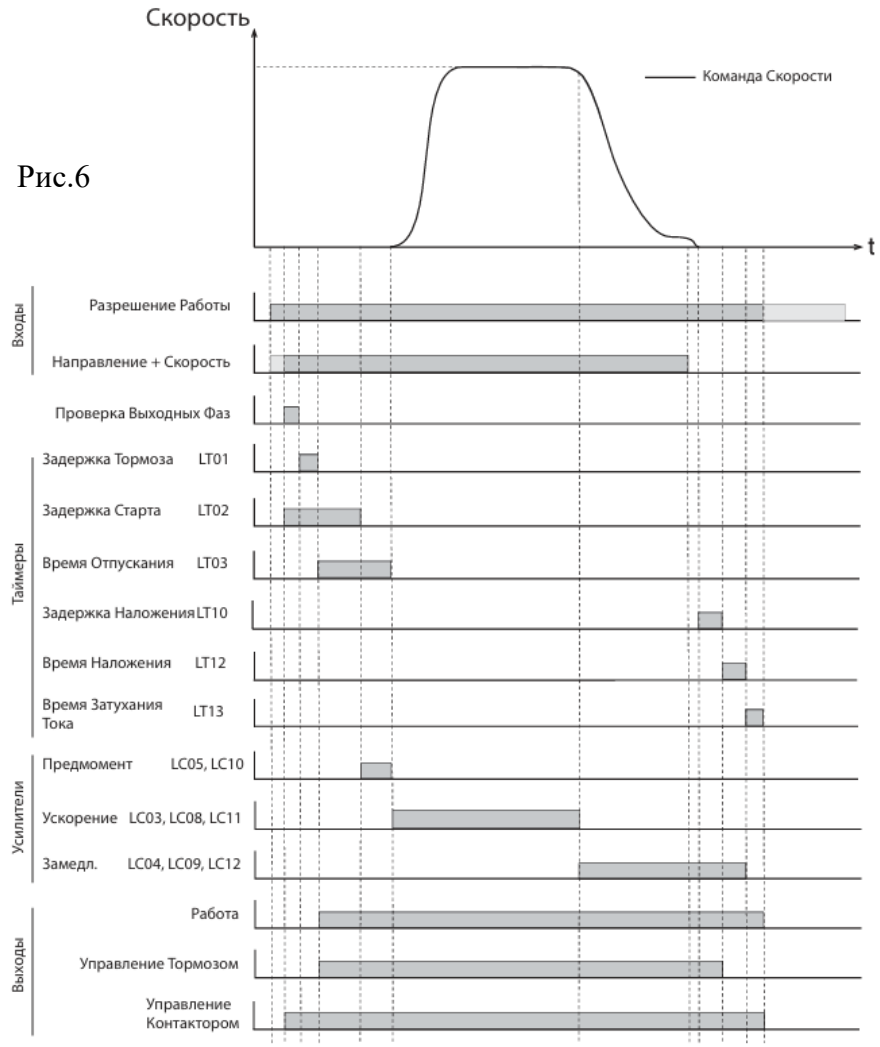
Пройдем по особенностям работы плат управления лифта. В системе управления лифтом, на самом деле, используется целый набор плат для разных взаимодействий с оборудованием.

Существуют: плата управления основная, плата на крыше кабины (для обработки сигналов управления движением в сервисном режиме, управлением приводом дверей, и обработке др. сигналов с датчиков), в приказной панели (для принятия приказов, и других команд), платы приема вызовов, плата взаимодействия с оборудованием в приямке. В дипломном проекте использован пример на работе тестирования основной платы управления, поскольку она является связующим звеном всего оборудования и на ней организован весь алгоритм работы. Взаимодействие с другими платами может быть реализовано обычным проводным соединением для получения нужных сигналов, либо передачей данных по специализированным протоколам соединения CANBUS, RTU Mod BUS или RS485. Каждый производитель, зачастую, создает свой специализированный набор команд передаваемых по шинам связи, хотя есть и универсальные протоколы. В проекте не учитывается возможности взаимодействия с платами по этим протоколам, однако следует учитывать это в дальнейшем, так можно получить некоторую информацию о состоянии системы, проходящих командах в системе.

В основе работы любой платы управления лифтом заложен алгоритм формирования управляющей последовательности выбора скоростей движения - для начала движения, движения на основной скорости и скоростей подхода к остановке, выравнивания в точной позиции. Типовой график движения в результате выбора скорости показан на рис. 5.

Рис.5

Рис.6



Частотный преобразователь работает в паре с платой управления – получает сигналы выбора скорости и формирует плавный пуск и остановку кабины лифта.

Пример диаграммы работы частотного преобразователя (KEB F5) показан на рис. 6.

В основном: плата выдает сигналы для частотного преобразователя разрешения и выбора скорости, а ЧП управляет контактором тормоза и обеспечивает плавные ускорения и замедления. Плата управления включает и выключает основной пускатель и другие устройства лифта (освещение и вентиляцию в кабине, приводы дверей).

Для корректной работы необходимо убедиться в том, что плата формирует правильную последовательность команд для ЧП и эти сигналы не пропадают во время движения. Также плата должна реагировать на смену режима работы, обрабатывать различные входные сигналы. В частности формирование последовательности команд зависит от входных сигналов, которые позволяют получить информацию о местоположении кабины в шахте. Во-первых: нужна информация, где находиться лифт - в верхней или нижней части шахты. Во-вторых: кабина находится в зоне остановки или нет. В-третьих: желательно понимать точное положение или пройденный путь между остановками. На рис. 8 показан пример размещения элементов для считывания положения кабины в шахте (возможный для рассматриваемый платы) с помощью магнитных датчиков.

Рис. 7

Моностабильные датчики ML1 ML2 служат для позиционирования кабины, определения точной остановки, срабатывают только в длинных магнитах длинной 30см.

Бистабильные датчики 817 и 818 для определения нахождения кабины вверху/внизу шахты. При движении вниз сработает датчик 817 и зафиксирует свое состояние до момента прохода кабины в обратном направлении – вверх, аналогично работает 818 датчик. Счетчик

пройденного пути кабиной реализуется с помощью подсчета импульсов от инкрементального (абсолютного) энкодера, размещенного на лебедке или на ограничителе скорости.

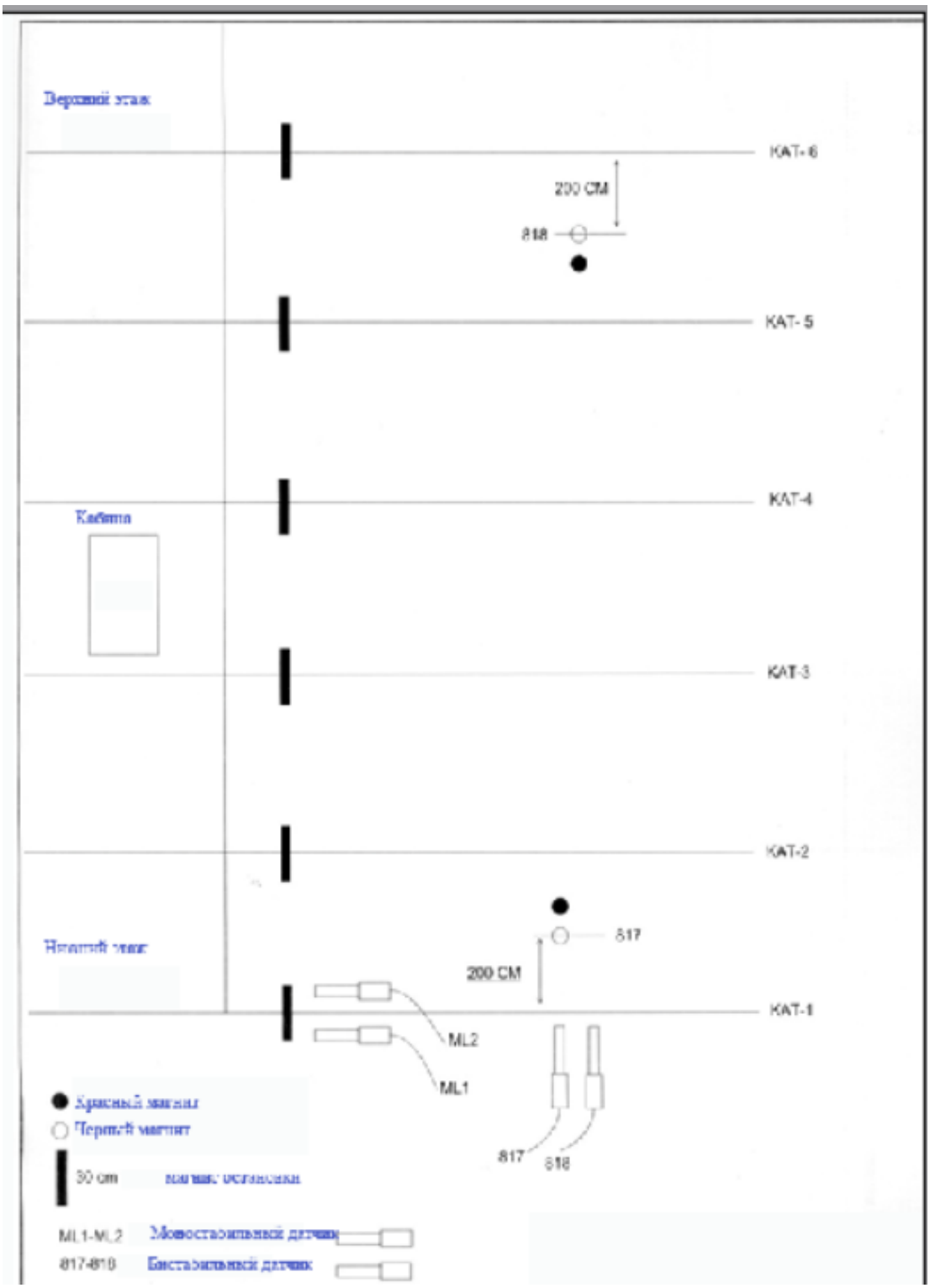


Рис. 8

В более старых системах расчет положения производился по времени пути движения кабины от контрольных точек, начиная от нижней остановки до этажа назначения.

Современные системы перед началом нормальной эксплуатации требуют произвести тестовую поездку для изучения расстояний между остановками и счета их количества, причем, необходимо завести в систему параметры лифта, такие как, например: скорость перемещения, количество остановок, значения коэффициентов подвески и диаметр шкива лебедки. Обучающая поездка часто делается в автоматическом режиме, во время которого плата производит расчет расстояний между остановками, а следовательно и необходимых расстояний для корректных замедлений. Работа энкодера представлена рис. 9

Четыре сигнала, состоящие из двух прямоугольных импульсов сдвинутых электрически на 90° по фазе и их инверсных сигналов. Минимальный уровень "1" соответствует напряжению 2,0В и максимальный уровень "0" соответствует напряжению 0,5В. Энкодер должен быть электрически изолирован от вала двигателя.

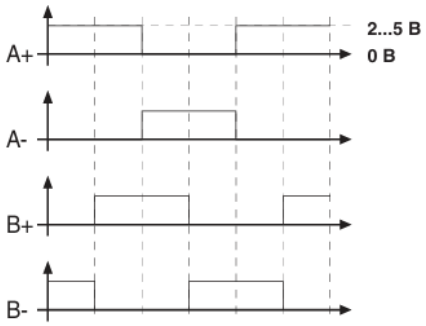


Рис. 9

Чтобы осуществить контроль за работой платы, необходим обработчик событий, который будет собирать информацию о состоянии платы, анализировать и отправлять результат анализа в качестве сообщений формата MQTT на сервер запущенный на микрокомпьютере RASPBERRY в системе Linux Ubuntu. К этому серверу можно будет подключиться дистанционно и просматривать состояние тестируемой платы. Все настройки для тестируемой платы будут храниться в настроечном файле.

Информация о событиях будет сохраняться в базе данных для дальнейшего анализа.

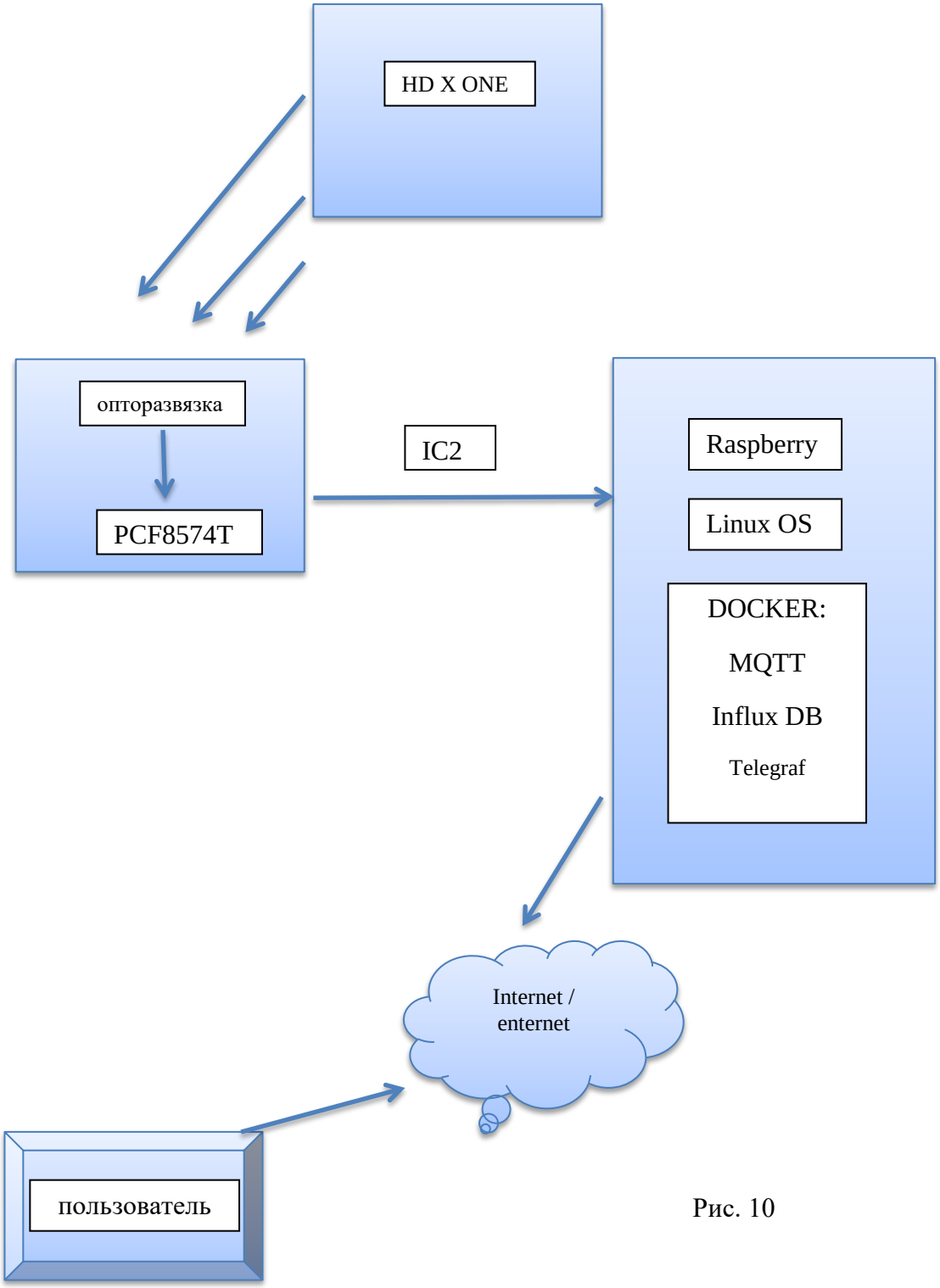


Рис. 10

MQTT или Message Queue Telemetry Transport – это легкий, компактный и открытый протокол обмена данными созданный для передачи данных на удалённых локациях, где требуется небольшой размер кода и есть ограничения по пропускной способности канала.

Брокер MQTT в основном отвечает за получение всех сообщений, отправленных по этому протоколу, их фильтрацию, определение получателей и передачу их конечным устройствам.

На Raspberry загрузим LINUX Ubuntu, установим сервер MQTT для приема сообщений и запустим программу – клиент для работы с входами и анализа обработки событий платы в нем же формируем сообщения для сервера MQTT. Необходимо установить сервер базы данных для хранения сообщений.

Устанавливаем MQTT paho.

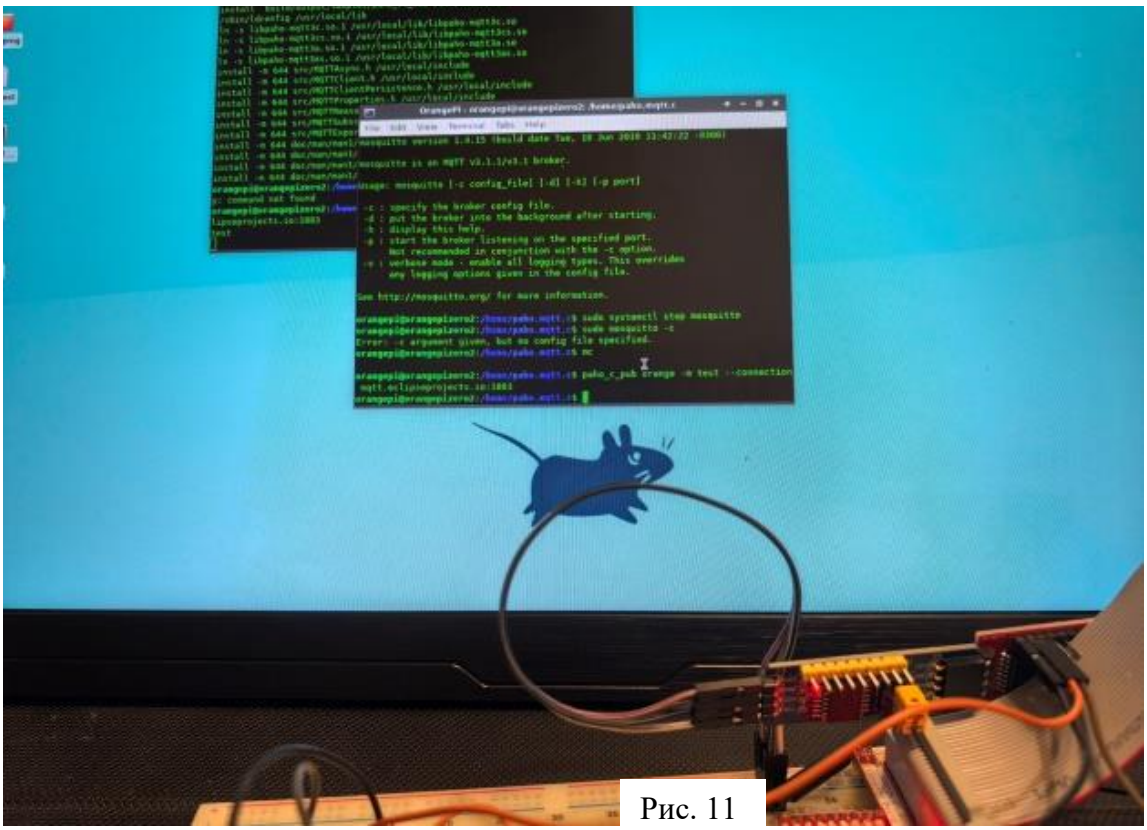


Рис. 11

Клонируем депозитарий с git

git clone https://github.com/eclipse/paho.mqtt.c.git

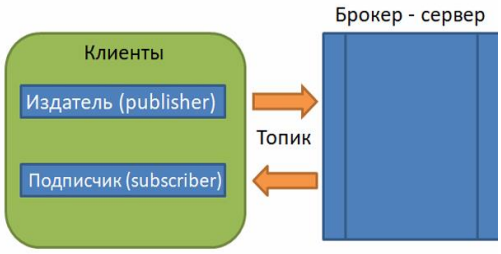


Рис. 12

переходим в папку mqtt

cd /home/paho.mqtt.c

Для установки пакета выполним команду:

sudo make install

Делаем проверку: открываем одно окно терминала и запускаем подписчика, а в другом окне терминала издателя с помощью команд:

настраиваем на прием сообщений:

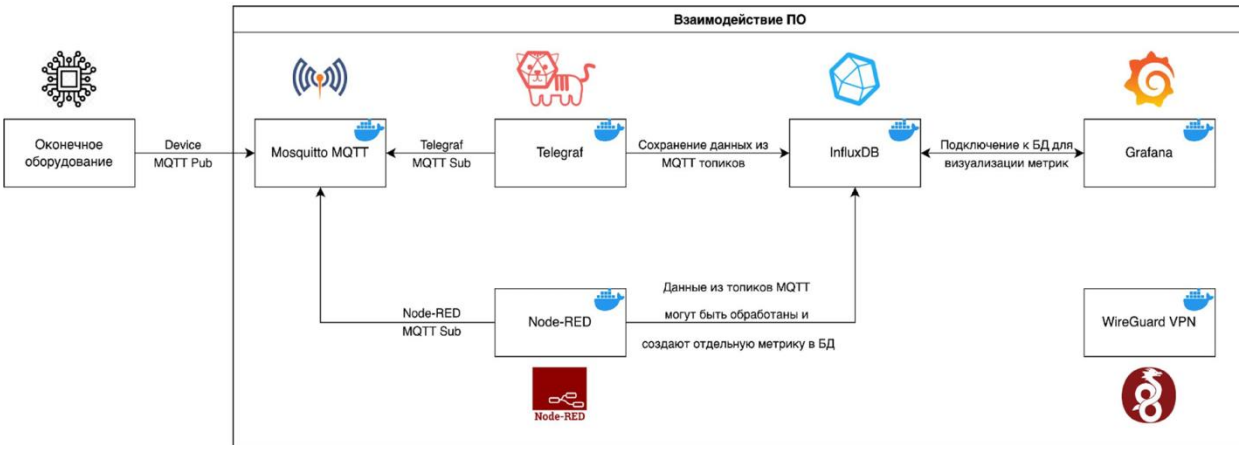
paho_c_sub orange --connection mqtt.eclipseprojects.io:1883

отправляем сообщение «test» для проверки

paho_c_pub orange –m test --connection mqtt.eclipseprojects.io:1883

Сообщение test проходит успешно (рис. 11).

Рис. 13



Использую материалы курса для создания оболочки для работы см. рис. 13.

Установим базу INFLUX DB и другое необходимое программное обеспечение.

sudo apt-get install influxdb

sudo apt-get install docker-compose

Можно запустить установку всех компонентов через файл конфигурации compose.yml для Docker путем запуска **install.sh** см. Приложение.

Docker позволяет легко инкапсулировать процесс создания и распространения программного продукта, чтобы можно было развернуть в произвольном масштабе, в любой среде, оптимизировать рабочие процессы, а так же таким образом увеличить скорость работы команд разработчиков. Docker вбирает в себя лучшее из самых эффективных технологий последнего десятилетия, позволяет заметно усовершенствовать почти каждый тип контейнера.

Поэтому, более удобно и предпочтительно, при развертывании системы сразу использовать Docker.

Создадим программу обработки данных для платформы Orange Pi.

Алгоритм работы программы обработчика:

- проверка соединения с платой расширения

- настройка платы на работу

- считывание состояния входов с платы по I2C

- декодирование значений считанного состояния:

выделение скорости, направления, состояний пускателей, датчиков 817 и 818, датчиков точной остановки ML1 и ML2, контроля тормоза, наличие цепи безопасности, состояний ревизии, пожара и др.

- передача состояния в MQTT, если состояние изменилось за интервал времени =1с.

- при аварийной ситуации – отправка специального сообщения с выводом на экран

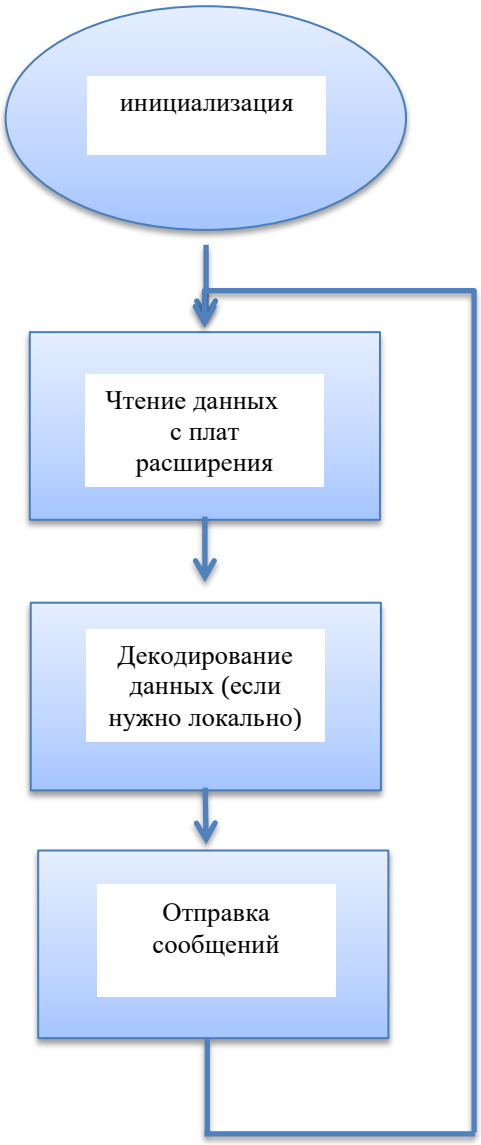


Рис. 14

Так как в проекте использовался клон Raspberry – Orange Pi используем библиотеку wiringOP https://github.com/orangepi-xunlong/wiringOP

*Node-RED — **инструмент программирования на основе потоков**, первоначально разработанный командой IBM Emerging Technology Services. Node-RED позволяет **подключать различные конечные точки**, такие как промышленные PLC, API, базы данных, корпоративные приложения и онлайн-сервисы.

Пользователи инструмента визуально перетаскивают «узлы» из палитры, которые представляют конечные точки, и затем соединяют их в потоки для выполнения желаемой задачи. Приложения Node-RED доступны через веб-браузер.

Node-RED используется в различных областях, включая **сбор данных с фабричного оборудования**, создание панелей визуализации данных, разработку платформ чат-ботов, извлечение, преобразование и интеграцию данных из разных источников.

Сделать декодирование сообщений можно также и в Node-Red добавив элементы “Decode” и “Function”. Диаграмма на рис. 15

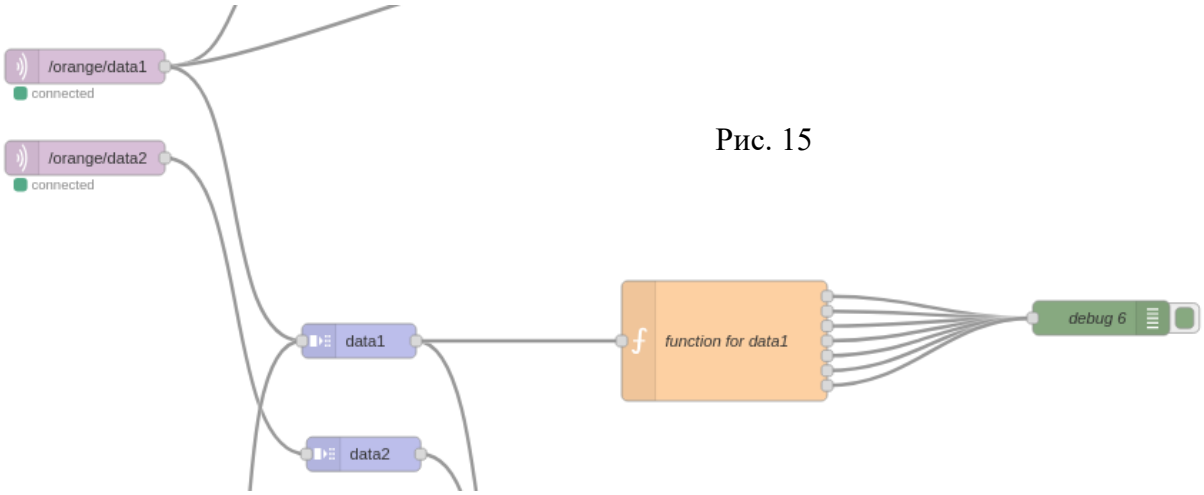


Рис. 15

Тестирование:

На собранном макете (рис. 17) проведем тестирование. Запускаем программу ph_mqtt_data_from_or.py. Текст программы для тестирования решения см. в приложении.

В ручном режиме управления платы начнем движение вверх,- запуститься реле включения основного контактора, выберется скорость 1 (RF включится) и направление вверх (RU1). Смотрим на экране в сообщениях на сервере Node-red (рис. 16).

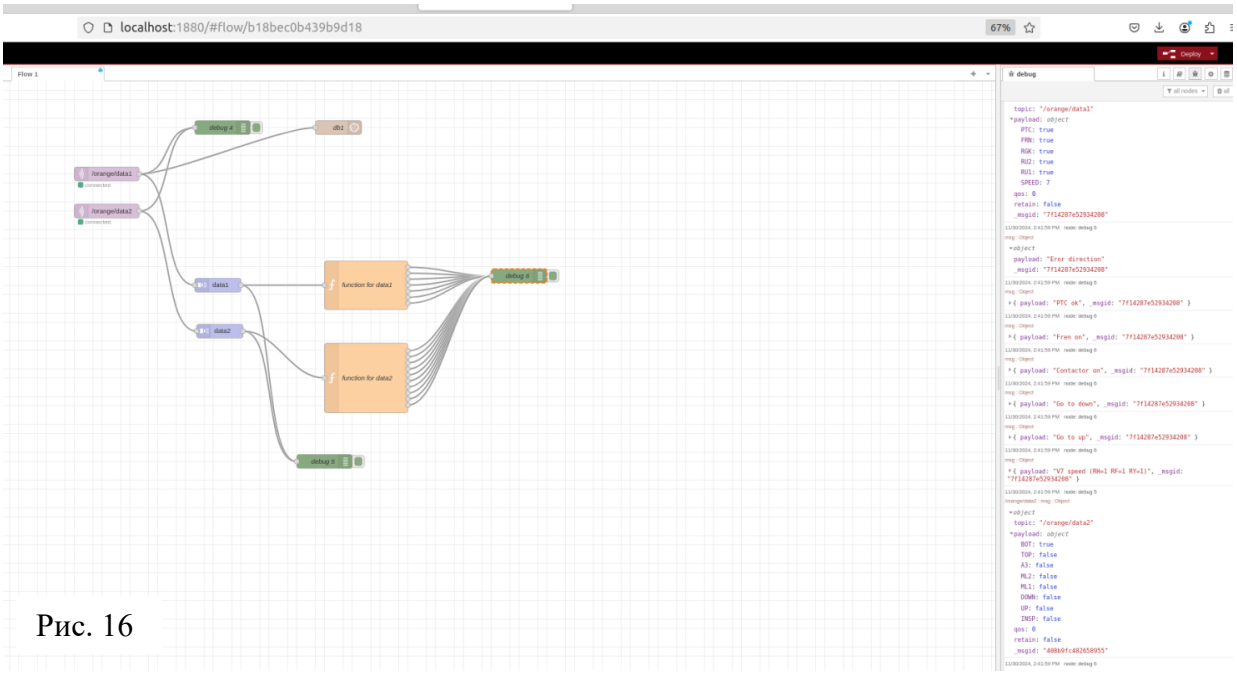

localhost:1880/#flow/b18bec0b439b9d18
Flow 1
/orange/data1
/orange/data2
data1
data2
function for data1
function for data2
debug
topic: "/orange/data1"
payload: object
PTC: true
FRN: true
RGK: true
RU2: true
RU1: true
SPEED: 7
qos: 0
retain: false
_msgid: "7f14287e52934208"
payload: "Error direction"
payload: "PTC ok"
payload: "Fren on"
payload: "Contactor on"
payload: "Go to down"
payload: "Go to up"
topic: "/orange/data2"
BOT: true
TOP: false
A3: false
ML2: false
ML1: false
DOWN: false
UP: false
INSP: false

Рис. 16

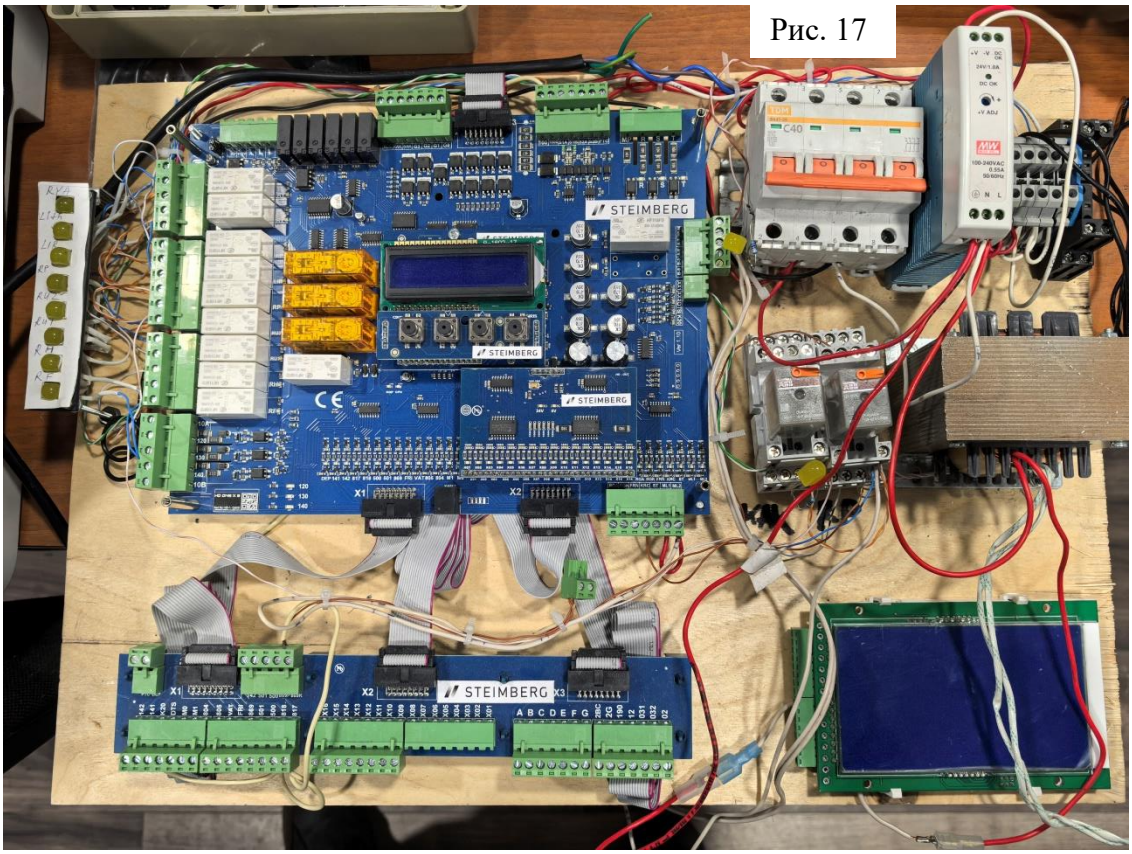

STEIMBERG
X1
X2
X3

Рис. 17

Заключение.

В современных реалиях очень важно следовать пути развития, замещения старого оборудования на новое, а это связано с обязательным контролем качества, поэтому необходимо создавать и развивать различное оборудование для тестирования и контроля.

Такое оборудование будет востребовано в сервисных центрах по обслуживанию лифтового оборудования, а так же в ремонтных специализированных мастерских.

В текущем проекте реализована базовая функция тестирования типовой платы в типовой ситуации.

В реальной реализации тестовое оборудование сможет симулировать входные сигналы, а затем контролировать реакцию платы. Будет доступна графическая версия с реализацией настроек в удобном для восприятия виде. В дальнейшем можно разрабатывать сценарии тестирования различных лифтовых плат с учетом их особенностей и необходимого режима работы для теста. Можно создавать блоки для расширения функций теста. Помимо этого необходимо создать удобный интерфейс для настройки и управления системы тестирования через web интерфейс для дистанционного взаимодействия с тестовой станцией при использовании круглосуточного тестирования на отказ оборудования.

Список используемой литературы:

Книга: «Raspberry Pi Руководство по настройке и применению» Магда Юрий Степанович. Москва, ДМК, 2014г.

Книга: «Практическая робототехника. C++ и RASPBERRY Pi» Ллойд Бромбах. Санкт-Петербург, BHV, 2023г.

Материалы курса IOT системы, https://gb.ru/

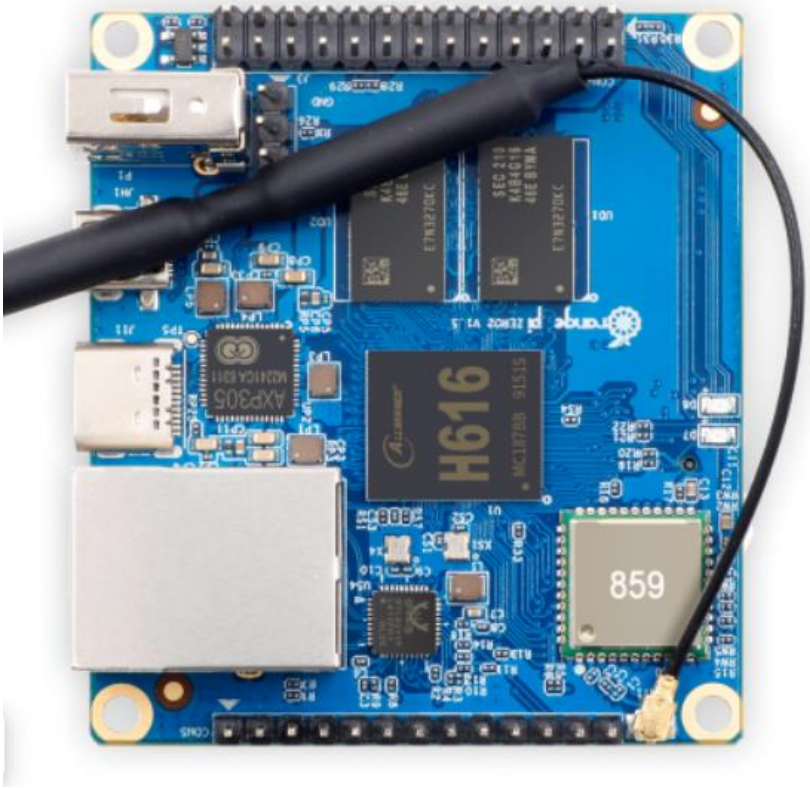
Инструкция по настройке и запуску лифта на базе платы Steimberg HD X one. Фролов А.Ю. 2015г.

Инструкция по настройке частотного преобразователя KEB F5, официальные материалы с сайта KEB.com.

Инструкция - “даташит” чипа PCF8574 материалы с сайта Texas Instrument www.ti.com.

Приложения

Характеристики Orange Pi Zero 2:



CPU	Allwinner H616 64-bit high-performance Quad-core Cortex-A53 processor
GPU	• Mali G31 MP2 • Supports OpenGL ES 1.0/2.0/3.2、OpenCL 2.0
Video decoding	• H.265 Main10@L5.1 decoder up to 4K@60fps or 6K@30fps • VP9 Profile 2 decoder up to 4K@60fps • AVS2 JiZhun 10bit decoder up to 4K@60fps • H.264 BP/MP/HP@L4.2 decoder up to 4K@30fps
Video encoding	• H.264 BP/MP/HP encoder up to 4K@25fps or 1080p@60fps • JPEG snapshot performance of 1080p@60fps
Memory(SDRAM)	512MB/1GB DDR3 (Shared with GPU）
Onboard Storage	• TF card slot • 2MB SPI Flash

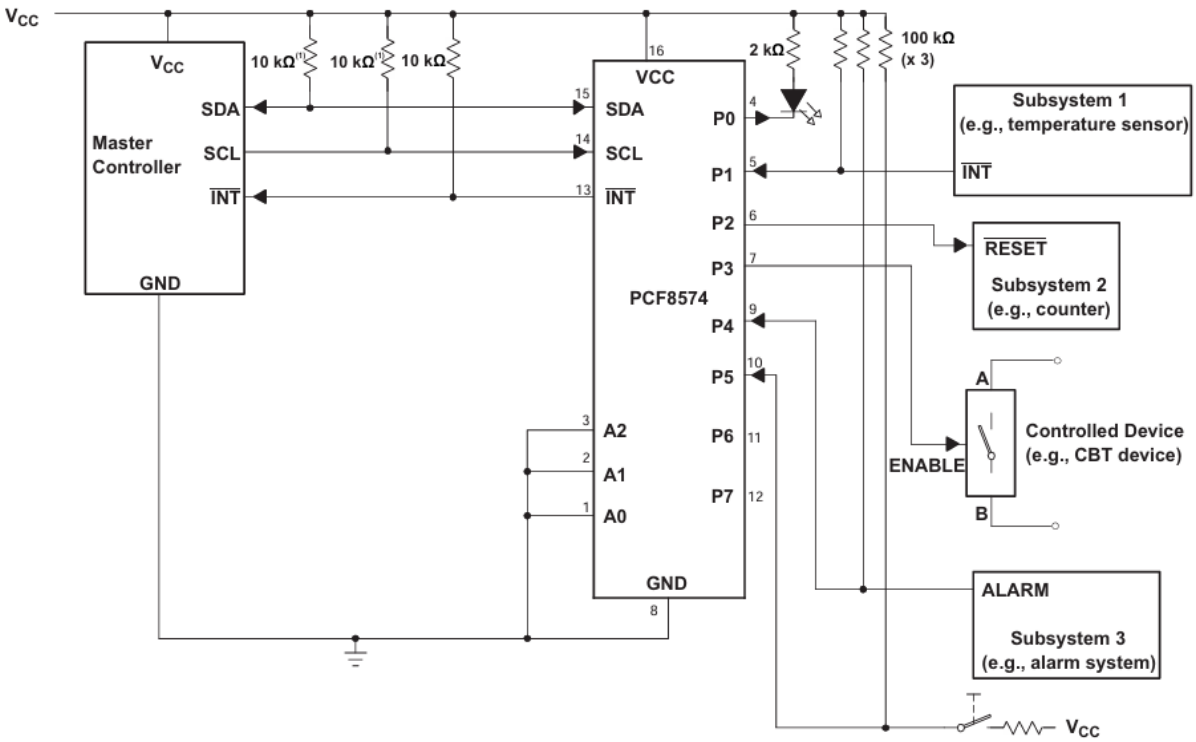
Onboard Network	Support 1000M/100M/10M Ethernet
Onboard WiFi+BT	• 859 Chip • Support IEEE 802.11 a/b/g/n/ac • Support BT5.0
Video Outputs	• Micro HDMI 2.0a up to 4K@60fps • TV CVBS output, Support PAL/NTSC （Via 13pin interface board）
Audio output	• Micro HDMI • 3.5mm audio port (Via 13pin interface board)
Power Source	Type-C interface 5V3A input
USB 2.0 Ports	3*USB 2.0 HOST（Two of them are via 13pin interface board）
Low-level peripherals	• 26pin header with I2C, SPI, UART and multiple GPIO ports • 13pin header with 2*USB Host, IR pin, Tv-out、AUDIO (no MIC) and 3 GPIO ports
Debug serial port	UART-TX、UART-RX and GND
LED	Power led & Status led
IR receiver	Support IR remote control (via 13pin interface board)
Supported OS	Android10、Ubuntu、Debian

Appearance specification introduction

Размеры	53mm×60mm
Вес	30g

Подключение для м/с PCF 8574 Texas Instrument

9.2 Typical Application



(1) The SCL and SDA pins must be tied directly to V_{CC} because if SCL and SDA are tied to an auxiliary power supply that could be powered on while V_{CC} is powered off, then the supply current, ICC, will increase as a result.

A. Device address is configured as 0100000 for this example.

B. P0, P2, and P3 are configured as outputs.

C. P1, P4, and P5 are configured as inputs.

D. P6 and P7 are not used and must be configured as outputs.

Файл обработки событий на микрокомпьютере Orange Pi zero 2 ph_mqtt_data_from_or.py

import smbus

import time

import paho.mqtt.publish as publish

bus= smbus.SMBus(3)

adr_2=0x24

adr_1=0x20

i=0

```
bus.write_byte(adr_2, 0xFF)
bus.write_byte(adr_1, 0xFF)
while (true):
        data1=bus.read_binary(adr_1))
        data1=bus.read_binary(adr_2))
        msgs = [{'topic': "/orange/data1", 'payload': data1}, ("/orange/data2", data2, 0, False)]
        publish.multiple(msgs, hostname="mqtt.eclipseprojects.io")
        sleep(1)
```

Настройки для функций в Node-Red

```
function data1
var new_msg=msg;
if (msg.payload.PTC==true)
{var ptc={payload: "PTC ok"};}
else
{var ptc = {payload: "PTC not ok"};}



switch (msg.payload.SPEED) {
    case 0:
        var speed = { payload: "0 speed" };
        break;
    case 1:
        var speed = { payload: "V1 speed (RH=1 RF=0 RY=0)" };
```

```
            break;
        case 2:
            var speed = { payload: "V2 speed (RH=0 RF=1 RY=0)" };
            break;
        case 3:
            var speed = { payload: "V3 speed (RH=1 RF=1 RY=0)" };
            break;
        case 4:
            var speed = { payload: "V4 speed (RH=0 RF=0 RY=1)" };
            break;
        case 5:
            var speed = { payload: "V5 speed (RH=1 RF=0 RY=1)" };
            break;
        case 6:
            var speed = { payload: "V6 speed (RH=0 RF=1 RY=1)" };
            break;
        case 7:
            var speed = { payload: "V7 speed (RH=1 RF=1 RY=1)" };
            break;
        default:
            var speed = { payload: "not correct speed" };
            break;
}
```

```
if (msg.payload.FRN == true) { var frn = { payload: "Fren on" }; }

else { var frn = { payload: "Fren off" }; }


if (msg.payload.RU1 == true) { var ru1 = { payload: "Go to up" }; }

if (msg.payload.RU2 == true) { var ru2 = { payload: "Go to down" }; }

if ((msg.payload.RU2 == true) && (msg.payload.RU1 == true )) { var err = { payload: "Eror direction" }; }

else { var err = { payload: "Ok" }; }


if (msg.payload.RGK == true) { var rgk = { payload: "Contactor on" }; }

else { var rgk = { payload: "Contactor off" }; }


return [err, ptc, frn, rgk, ru2, ru1, speed];


function data2


var new_msg=msg;


if (msg.payload.BOT==false) {var pit={payload: "nearly Pit"};}

if (msg.payload.TOP == false) { var top = { payload: "nearly Top" }; }


if ((msg.payload.TOP == false) && (msg.payload.BOT == false)) { var err1 = { payload: "Eror top and bot together" }; }

else { var err1 = { payload: " " }; }
```

```
if (msg.payload.A3 == true) { var opcl = { payload: "door is close" }; }

else { var opcl = { payload: "door is open" }; }


if (msg.payload.ML2 == true) { var ml2 = { payload: "ML2 is on" }; }

else { var ml2 = { payload: "" }; }


if (msg.payload.ML1 == true) { var ml1 = { payload: "ML1 is on" }; }

else { var ml1 = { payload: "" }; }


if ((msg.payload.ML2 == true) && (msg.payload.ML1 == true)) { var ml2 = { payload: "In floor" }; var ml1 = { payload: "In floor" }; }

if (msg.payload.DOWN == true) { var dwn = { payload: "move down" }; }

else { var dwn = { payload: "" }; }

if (msg.payload.UP == true) { var up = { payload: "move up" }; }

else { var up = { payload: "" }; }

if (msg.payload.INSP == true) { var insp = { payload: "Normal" }; }

else { var insp = { payload: "Inspection" }; }

if(((new_msg.payload.UP == true) || (new_msg.payload.DOWN == true)) && (new_msg.payload.INSP == true) )

   {var err2 = { payload: "Try Move insp in normal" }; }

if (((new_msg.payload.UP == true) && (new_msg.payload.DOWN == true)) && (new_msg.payload.INSP == false))

   { var err3 = { payload: "error Move in insp both side" }; }

return [err1, err2, err3, pit, top, opcl, ml2, ml1, dwn, up, insp];
```

Файл конфигурации для Node-Red node_red_flow.json


```
[
    {
        "id": "b18bec0b439b9d18",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "5f0269884354e473",
        "type": "mqtt in",
        "z": "b18bec0b439b9d18",
        "name": "",
        "topic": "/orange/data1",
        "qos": "1",
        "datatype": "auto-detect",
        "broker": "3ef8ed1b788e6706",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 230,
```

```
    "y": 200,
    "wires": [
      [
        "f9f1589a13535bd9",
        "d8d8d4322e81c2c7",
        "adbf2ebcde129379"
      ]
    ]
  },
  {
    "id": "f9f1589a13535bd9",
    "type": "debug",
    "z": "b18bec0b439b9d18",
    "name": "debug 4",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 480,
    "y": 100,
    "wires": []
```

```
},
{
    "id": "adbf2ebcde129379",
    "type": "bit-decode",
    "z": "b18bec0b439b9d18",
    "name": "data1",
    "bits": "3e09b2ac37fd8b79",
    "x": 470,
    "y": 440,
    "wires": [
      [
        "2db0875c6ac8b5c9",
        "6d0da4179346d562"
      ]
    ]
},
{
    "id": "d8d8d4322e81c2c7",
    "type": "influxdb out",
    "z": "b18bec0b439b9d18",
    "influxdb": "9aca28e143868018",
    "name": "db1",
    "measurement": "",
    "precision": "",
```

```
    "retentionPolicy": "",
    "database": "database",
    "precisionV18FluxV20": "ms",
    "retentionPolicyV18Flux": "",
    "org": "organisation",
    "bucket": "bucket",
    "x": 730,
    "y": 100,
    "wires": []
},
{
    "id": "342f61db9f064ab2",
    "type": "mqtt in",
    "z": "b18bec0b439b9d18",
    "name": "",
    "topic": "/orange/data2",
    "qos": "1",
    "datatype": "auto-detect",
    "broker": "3ef8ed1b788e6706",
    "nl": false,
    "rap": true,
    "rh": 0,
    "inputs": 0,
    "x": 230,
```

```
    "y": 280,
    "wires": [
      [
        "660e456677d5289d",
        "f9f1589a13535bd9"
      ]
    ]
  },
  {
    "id": "2db0875c6ac8b5c9",
    "type": "debug",
    "z": "b18bec0b439b9d18",
    "name": "debug 5",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "true",
    "targetType": "full",
    "statusVal": "",
    "statusType": "auto",
    "x": 700,
    "y": 820,
    "wires": []
```

```
},
{
    "id": "6d0da4179346d562",
    "type": "function",
    "z": "b18bec0b439b9d18",
    "name": "function for data1",
    "func": "var new_msg=msg;\nif (msg.payload.PTC==true)\n{var ptc={payload: \"PTC ok\"};}\nelse \n{var ptc = {payload: \"PTC not ok\"};}\n\n\nswitch (msg.payload.SPEED) {\n case 0:\n var speed = { payload: \"0 speed\" };\n break;\n case 1:\n var speed = { payload: \"V1 speed (RH=1 RF=0 RY=0)\" };\n break;\n case 2:\n var speed = { payload: \"V2 speed (RH=0 RF=1 RY=0)\" };\n break;\n case 3:\n var speed = { payload: \"V3 speed (RH=1 RF=1 RY=0)\" };\n break;\n case 4:\n var speed = { payload: \"V4 speed (RH=0 RF=0 RY=1)\" };\n break;\n case 5:\n var speed = { payload: \"V5 speed (RH=1 RF=0 RY=1)\" };\n break;\n case 6:\n var speed = { payload: \"V6 speed (RH=0 RF=1 RY=1)\" };\n break;\n case 7:\n var speed = { payload: \"V7 speed (RH=1 RF=1 RY=1)\" };\n break;\n default:\n var speed = { payload: \"not correct speed\" };\n break;\n} \n\nif (msg.payload.FRN == true) { var frn = { payload: \"Fren on\" }; }\nelse { var frn = { payload: \"Fren off\" }; }\n\nif (msg.payload.RU1 == true) { var ru1 = { payload: \"Go to up\" }; }\nif (msg.payload.RU2 == true) { var ru2 = { payload: \"Go to down\" }; }\nif ((msg.payload.RU2 == true) && (msg.payload.RU1 == true )) { var err = { payload: \"Eror direction\" }; }\nelse { var err = { payload: \"Ok\" }; }\n\nif (msg.payload.RGK == true) { var rgk = { payload: \"Contactor on\" }; }\nelse { var rgk = { payload: \"Contactor off\" }; }\n\nreturn [err, ptc, frn, rgk, ru2, ru1, speed];",
    "outputs": 7,
    "timeout": 0,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
```

```
"libs": [],
"x": 790,
"y": 440,
"wires": [
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
    ],
    [
        "95fe5c98a92b1857"
```

```
        ]
    ]
},
{
    "id": "660e456677d5289d",
    "type": "bit-decode",
    "z": "b18bec0b439b9d18",
    "name": "data2",
    "bits": "4f9082451664583c",
    "x": 474,
    "y": 539,
    "wires": [
        [
            "2db0875c6ac8b5c9",
            "d49ce28ba13d1a5d"
        ]
    ]
},
{
    "id": "95fe5c98a92b1857",
    "type": "debug",
    "z": "b18bec0b439b9d18",
    "name": "debug 6",
    "active": true,
```

```
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "true",
        "targetType": "full",
        "statusVal": "",
        "statusType": "auto",
        "x": 1120,
        "y": 420,
        "wires": []
    },
    {
        "id": "d49ce28ba13d1a5d",
        "type": "function",
        "z": "b18bec0b439b9d18",
        "name": "function for data2",
        "func": "var new_msg=msg;\n\nif (msg.payload.BOT==false) {var pit={payload: \"nearly Pit\"};}\nif (msg.payload.TOP == false) { var top = { payload: \"nearly Top\" }; }\n\nif ((msg.payload.TOP == false) && (msg.payload.BOT == false)) { var err1 = { payload: \"Eror top and bot together\" }; }\nelse { var err1 = { payload: \" \" }; } \n\nif (msg.payload.A3 == true) { var opcl = { payload: \"door is close\" }; }\nelse { var opcl = { payload: \"door is open\" }; }\n\nif (msg.payload.ML2 == true) { var ml2 = { payload: \"ML2 is on\" }; }\nelse { var ml2 = { payload: \"\" }; }\n\nif (msg.payload.ML1 == true) { var ml1 = { payload: \"ML1 is on\" }; }\nelse { var ml1 = { payload: \"\" }; }\n\nif ((msg.payload.ML2 == true) && (msg.payload.ML1 == true)) { var ml2 = { payload: \"In floor\" }; var ml1 = { payload: \"In floor\" }; }\n\nif (msg.payload.DOWN == true) { var dwn = { payload: \"move down\" }; }\nelse { var dwn = { payload: \"\" }; }\n\nif (msg.payload.UP == true) { var up = {
```

payload: \"move up\" }; }\nelse { var up = { payload: \"\" }; }\n\nif (msg.payload.INSP == true) { var insp = { payload: \"Normal\" }; }\nelse { var insp = { payload: \"Inspection\" }; }\n\nif(((new_msg.payload.UP == true) || (new_msg.payload.DOWN == true)) && (new_msg.payload.INSP == true))\n {var err2 = { payload: \"Try Move insp in normal\" }; }\n\nif (((new_msg.payload.UP == true) && (new_msg.payload.DOWN == true)) && (new_msg.payload.INSP == false))\n { var err3 = { payload: \"error Move in insp both side\" }; }\n\nreturn [err1, err2, err3, pit, top, opcl, ml2, ml1, dwn, up, insp];",

"outputs": 10,

"timeout": 0,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 790,

"y": 640,

"wires": [

[

"95fe5c98a92b1857"

],

[

"95fe5c98a92b1857"

],

[

"95fe5c98a92b1857"

],

[

```
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ],
        [
            "95fe5c98a92b1857"
        ]
    ]
},
{
    "id": "3ef8ed1b788e6706",
```

"type": "mqtt-broker",

"name": "orange",

"broker": "mqtt.eclipseprojects.io",

"port": "1883",

"clientid": "Orangepi",

"autoConnect": true,

"usetls": false,

"protocolVersion": "4",

"keepalive": "60",

"cleansession": true,

"autoUnsubscribe": true,

"birthTopic": "",

"birthQos": "1",

"birthRetain": "false",

"birthPayload": "",

"birthMsg": {},

"closeTopic": "",

"closeQos": "0",

"closeRetain": "false",

"closePayload": "",

"closeMsg": {},

"willTopic": "",

"willQos": "0",

"willRetain": "false",

```
    "willPayload": "",
    "willMsg": {},
    "userProps": "",
    "sessionExpiry": ""
},
{
    "id": "3e09b2ac37fd8b79",
    "type": "bit-config",
    "name": "move",
    "bits": [
        {
            "property": "PTC",
            "length": 1,
            "shift": 0,
            "type": "boolean"
        },
        {
            "property": "FRN",
            "length": 1,
            "shift": 1,
            "type": "boolean"
        },
        {
            "property": "RGK",
```

```
            "length": 1,
            "shift": 2,
            "type": "boolean"
        },
        {
            "property": "RU2",
            "length": 1,
            "shift": 3,
            "type": "boolean"
        },
        {
            "property": "RU1",
            "length": 1,
            "shift": 4,
            "type": "boolean"
        },
        {
            "property": "SPEED",
            "length": 3,
            "shift": 5,
            "type": "number"
        }
    ]
},
```

```
{
    "id": "9aca28e143868018",
    "type": "influxdb",
    "hostname": "127.0.0.1",
    "port": "8086",
    "protocol": "http",
    "database": "database",
    "name": "db1",
    "usetls": false,
    "tls": "",
    "influxdbVersion": "1.x",
    "url": "http://localhost:8086",
    "timeout": "10",
    "rejectUnauthorized": true
},
{
    "id": "4f9082451664583c",
    "type": "bit-config",
    "name": "status",
    "bits": [
      {
        "property": "BOT",
        "length": 1,
        "shift": 0,
```

```
    "type": "boolean"
},
{
    "property": "TOP",
    "length": 1,
    "shift": 1,
    "type": "boolean"
},
{
    "property": "A3",
    "length": 1,
    "shift": 2,
    "type": "boolean"
},
{
    "property": "ML2",
    "length": 1,
    "shift": 3,
    "type": "boolean"
},
{
    "property": "ML1",
    "length": 1,
    "shift": 4,
```

```
            "type": "boolean"
        },
        {
            "property": "DOWN",
            "length": 1,
            "shift": 5,
            "type": "boolean"
        },
        {
            "property": "UP",
            "length": 1,
            "shift": 6,
            "type": "boolean"
        },
        {
            "property": "INSP",
            "length": 1,
            "shift": 7,
            "type": "boolean"
        }
    ]
  }
]
```

Файл install2.sh

 # Создаем скрипт автоматической установки Docker и необходимых файлов конфигураций nano install2.sh

#!/bin/bash

 # Получаем IP адрес машины, на которой запускается скрипт

vm_ip=$(ip a | awk '/inet / && !/127.0.0.1/ {gsub(/\/.*/, "", $2); print $2; exit}')

echo "IP Машины: "$vm_ip

#Проверяем, является ли пользователь root'ом

if ["$EUID" -eq 0]; then

echo "Этот скрипт не должен запускаться с правами root. Запустите его от имени обычного пользователя."

exit 1

fi

 # Получаем имя текущего пользователя

current_user=$USER

 # Создаем необходимые каталоги для docker-compose.yml

user_home=$(eval echo ~$current_user)

mkdir -p $user_home

 # Генерируем уникальные порты для пользователя

mosquitto_port=1883

influxdb_port=8086

grafana_port=3000

nodered_port=1880

wireguard_port=51820

telegraf_port1=8092

telegraf_port2=8094

telegraf_port3=8125

nginx_port1=80

nginx_port2=443

 # Задаем доменное имя для конфигурации nginx без www. и без .ru

nginx_domain=orangepi

read -p "Введите доменное имя для конфигурации nginx без www. и без .ru: " nginx_domain

read -p "Введите среднюю часть доменного имени для ваших сервисов из инициалов и номера группы, например, gvi-4538 (для grafana-gvi-4538.gb-iot.ru и др.): " nginx_3rd_domain

 # Задаем данные для подключения сервиса DDNS No-IP

#echo -n "Введите username с сервиса NO-IP.com : "

#read inadyn_username

#echo -n "Введите password с сервиса NO-IP.com : "

#read -s inadyn_password

#echo

#echo -n "Введите hostname с сервиса NO-IP.com : "

#read inadyn_hostname

 # Создаем docker-compose.yml для текущего пользователя

cat > "$user_home/docker-compose.yml" << EOF

version: '3'

services:

mosquitto:

```
  image: eclipse-mosquitto
  network_mode: bridge
  user: $(id -u $current_user):$(id -g $current_user)
  ports:
    - "$mosquitto_port:1883"
  volumes:
    - "$user_home/mosquitto/config:/mosquitto/config"
    - "$user_home/mosquitto/data:/mosquitto/data"
    - "$user_home/mosquitto/log:/mosquitto/log"
  environment:
    - TZ=Europe/Moscow
  restart: unless-stopped
telegraf:
  image: telegraf:alpine
  network_mode: bridge
  user: $(id -u $current_user):$(id -g $current_user)
  volumes:
    - "$user_home/telegraf/conf:/etc/telegraf/telegraf.d"
  ports:
    - "$telegraf_port1:8092/tcp"
    - "$telegraf_port1:8092/udp"
    - "$telegraf_port2:8094/tcp"
    - "$telegraf_port2:8094/udp"
    - "$telegraf_port3:8125/tcp"
```

```
    - "$telegraf_port3:8125/udp"
  environment:
    - TZ=Europe/Moscow
  depends_on:
    - mosquitto
    - influxdb
  restart: unless-stopped
influxdb:
  image: influxdb:alpine
  network_mode: bridge
  user: $(id -u $current_user):$(id -g $current_user)
  ports:
    - "$influxdb_port:8086"
  volumes:
    - "$user_home/influxdb/data:/var/lib/influxdb2"
    - "$user_home/influxdb/conf:/etc/influxdb"
    - "$user_home/influxdb/engine:/var/lib/influxdb2/engine"
  environment:
    - TZ=Europe/Moscow
  restart: unless-stopped
grafana:
  image: grafana/grafana
  network_mode: bridge
  user: $(id -u $current_user):$(id -g $current_user)
```

```
  volumes:
    - "$user_home/grafana/data:/var/lib/grafana"
    - "$user_home/grafana/conf:/etc/grafana"
    - "$user_home/grafana/log:/var/log/grafana"
  ports:
    - "$grafana_port:3000"
  environment:
    - PUID=$(id -u $current_user)
    - PGID=$(id -g $current_user)
    - TZ=Europe/Moscow
  restart: unless-stopped
nodered:
  image: nodered/node-red:latest-minimal
  network_mode: bridge
  user: $(id -u $current_user):$(id -g $current_user)
  volumes:
    - "$user_home/node-red/data:/data"
  ports:
    - "$nodered_port:1880"
  environment:
    - TZ=Europe/Moscow
  restart: unless-stopped
wireguard:
  image: lscr.io/linuxserver/wireguard:latest
```

network_mode: bridge

privileged: true

volumes:

- "$user_home/wireguard/config:/config"

ports:

- "$wireguard_port:51820/tcp"

- "$wireguard_port:51820/udp"

environment:

- PUID=$(id -u $current_user)

- PGID=$(id -g $current_user)

- TZ=Europe/Moscow

- SERVERURL=$vm_ip #optional

- SERVERPORT=$wireguard_port #optional

- PEERS=10 #optional

- PEERDNS=1.1.1.1 #optional

- INTERNAL_SUBNET=10.13.13.0 #optional

- ALLOWEDIPS=0.0.0.0/0 #optional

- LOG_CONFS=false #optional

cap_add:

- NET_ADMIN

- SYS_MODULE

sysctls:

- net.ipv4.conf.all.src_valid_mark=1

restart: unless-stopped

```
  nginx:
    image: nginx:mainline-alpine-slim
    network_mode: bridge
    volumes:
      - "$user_home/nginx/conf:/etc/nginx/conf.d"
      - "$user_home/nginx/html:/usr/share/nginx/html"
      - "$user_home/certbot/conf:/etc/letsencrypt"
      - "$user_home/certbot/www:/var/www/certbot"
    environment:
      - TZ=Europe/Moscow
    ports:
      - "$nginx_port1:80"
      - "$nginx_port2:443"
    command: '/bin/sh -c ''while :; do sleep 24h & wait \$\${!}; nginx -s reload; done & nginx -g
"daemon off;"''''
    restart: unless-stopped
  certbot:
    image: certbot/certbot:latest
    network_mode: bridge
    volumes:
      - "$user_home/certbot/conf:/etc/letsencrypt"
      - "$user_home/certbot/www:/var/www/certbot"
    environment:
      - TZ=Europe/Moscow
```

```
    depends_on:
      - nginx
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 24h & wait \$\${!}; done;'"
    restart: unless-stopped
  inadyn:
    image: troglobit/inadyn:latest
    restart: unless-stopped
    network_mode: bridge
    volumes:
      - "$user_home/inadyn/inadyn.conf:/etc/inadyn.conf"
    environment:
      - TZ=Europe/Moscow
EOF
mkdir -m 755 -p $user_home/nginx/conf
mkdir -m 755 -p $user_home/nginx/html
mkdir -m 755 -p $user_home/certbot/conf
mkdir -m 755 -p $user_home/certbot/www
mkdir -m 755 -p $user_home/inadyn
# Создаем файлы конфигурации для inadyn для текущего пользователя
cat > "$user_home/inadyn/inadyn.conf" << EOF
# In-A-Dyn v2.0 configuration file format
period = 60
user-agent = Mozilla/5.0
```

```
provider no-ip.com {
   username    = $inadyn_username
   password    = $inadyn_password
   hostname    = $inadyn_hostname
}
EOF

# Создаем файлы конфигурации для NGINX для текущего пользователя
cat > "$user_home/nginx/conf/nginx.conf" << EOSF
server {
   listen 80;
   server_name $nginx_3rd_domain.$nginx_domain.ru;
   return 301 https://$nginx_3rd_domain.$nginx_domain.ru\$request_uri;
}
server {
   listen 80;
   server_name $nginx_3rd_domain.$nginx_domain.ru;
   root /usr/share/nginx/html/$nginx_domain.ru;
   index index.html;
   location / {
      try_files \$uri \$uri/ /index.html;
   }
}
server {
```

```
    listen 443 ssl http2;
    server_name $nginx_3rd_domain.$nginx_domain.ru;
    root /usr/share/nginx/html/$nginx_domain.ru;
    ssl_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/$nginx_domain.ru/privkey.pem;


    index index.html;
    location / {
        try_files \$uri \$uri/ /index.html;
    }
}
##################################################################
# Grafana primary listener and redirect
server {
  listen 80;
  server_name grafana-$nginx_3rd_domain.$nginx_domain.ru;
  return 301 https://\$host\$request_uri;
}
# Grafana ssl config
server {
  listen 443 ssl http2;
  server_name grafana-$nginx_3rd_domain.$nginx_domain.ru;
  ssl_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
```

```
  ssl_trusted_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/$nginx_domain.ru/privkey.pem;
  location / {
  http2_push_preload on;
  proxy_pass http://$vm_ip:$grafana_port;
  proxy_set_header Host \$host;
  proxy_set_header Connection "upgrade";
  proxy_set_header Upgrade \$http_upgrade;
  proxy_set_header X-Real-IP \$remote_addr;
  proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
  add_header X-Http-Version \$server_protocol;
  proxy_buffering on;
  proxy_buffer_size 8k;
  proxy_buffers 2048 8k;
  }
}
##########################################################################
# Influxdb primary listener and redirect
server {
  listen 80;
  server_name influxdb-$nginx_3rd_domain.$nginx_domain.ru;
  return 301 https://\$host\$request_uri;
}
# Influxdb ssl config
```

```
server {
  listen 443 ssl http2;
  server_name influxdb-$nginx_3rd_domain.$nginx_domain.ru;
  ssl_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
  ssl_trusted_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/$nginx_domain.ru/privkey.pem;

  location / {
  http2_push_preload on;
  proxy_pass http://$vm_ip:$influxdb_port;
  proxy_set_header Host \$host;
  proxy_set_header Connection "upgrade";
  proxy_set_header Upgrade \$http_upgrade;
  proxy_set_header X-Real-IP \$remote_addr;
  proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
  add_header X-Http-Version \$server_protocol;
  proxy_buffering on;
  proxy_buffer_size 8k;
  proxy_buffers 2048 8k;
  }
}
##########################################################################
# Nodered primary listener and redirect
server {
```

```
  listen 80;
  server_name nodered-$nginx_3rd_domain.$nginx_domain.ru;
  return 301 https://\$host\$request_uri;
}
# Nodered ssl config
server {
  listen 443 ssl http2;
  server_name nodered-$nginx_3rd_domain.$nginx_domain.ru;
  ssl_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
  ssl_trusted_certificate /etc/letsencrypt/live/$nginx_domain.ru/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/$nginx_domain.ru/privkey.pem;
  location / {
  http2_push_preload on;
  proxy_pass http://$vm_ip:$nodered_port;
  proxy_set_header Host \$host;
  proxy_set_header Connection "upgrade";
  proxy_set_header Upgrade \$http_upgrade;
  proxy_set_header X-Real-IP \$remote_addr;
  proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
  add_header X-Http-Version \$server_protocol;
  proxy_buffering on;
  proxy_buffer_size 8k;
  proxy_buffers 2048 8k;
  }
```

```
}

EOSF

# Создаем скрипт для настройки пользовательских параметров

cat > "$user_home/settings2.sh" << EOF

#!/bin/bash

docker exec -it $current_user-certbot-1 sh -c 'certbot -d *.$nginx_domain.ru -d $nginx_domain.ru --manual --preferred-challenges dns certonly --server https://acme-v02.api.letsencrypt.org/directory'

EOF

#Скачивается папка с сайтом-заглушкой и названием домена в папку /nginx/html/

#wget https://codeload.github.com/VladislavGatsenko/gb-iot/zip/refs/heads/main -O #$nginx_domain.zip

#unzip $nginx_domain.zip -d $user_home/nginx/html/

#mv "$user_home/nginx/html/"* "$user_home/nginx/html/$nginx_domain.ru"

# Устанавливаем владельца и разрешения для созданных файлов конфигурации

chown $current_user:$current_user "$user_home/docker-compose.yml"

chown $current_user:$current_user "$user_home/settings2.sh"

chown $current_user:$current_user "$user_home/nginx/conf/nginx.conf"

chown $current_user:$current_user "$user_home/inadyn/inadyn.conf"

chown $current_user:$current_user "$user_home/nginx/html/$nginx_domain.ru"

chmod 755 "$user_home/docker-compose.yml"

chmod 755 "$user_home/settings2.sh"

chmod 755 "$user_home/nginx/conf/nginx.conf"

chmod 755 "$user_home/inadyn/inadyn.conf"

chmod 755 "$user_home/nginx/html/$nginx_domain.ru"
```

echo

echo "Перед запуском Nginx не забыть вставить сертификаты (если они есть) в папку $user_home/certbot/conf/live/$nginx_domain.ru . Если их нет, они будут сгенерированы позже командой bash settings2.sh"

---

bash settings2.sh

docker compose down

docker compose up -d