Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Мобильные вычислительные системы»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководители курсового проекта
Доцент кафедры ПИКС
_____В.С. Колбун
__.__.2022
Доцент кафедры ПИКС
_____О.Ч. Ролич
__.__.2022

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
на тему:
**«МОБИЛЬНАЯ *IP*-ВИДЕОКАМЕРА»**

БГУИР КП 1-39 03 02 045 ПЗ

Выполнил студент группы 913801
Гончар Виталий Витальевич

_____
(подпись студента)
Курсовой проект представлен на проверку ___.____.2022

_____
(подпись студента)

Минск 2022

# РЕФЕРАТ


БГУИР КП 1-39 03 02 045 ПЗ

Гончар, В.В. Мобильная *IP*-видеокамера: пояснительная записка к курсовому проекту / В.В. Гончар. – Минск: БГУИР, 2022. – 95 с.

Пояснительная записка 95 с., 26 рисунков, 40 источников, 5 приложений.

АНАЛИЗ МЕТОДОВ ПРОЕКТИРОВАНИЯ МОБИЛЬНОЙ *IP*-ВИДЕОКАМЕРЫ, ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА КОНСТРУКЦИИ МОБИЛЬНОЙ *IP*-ВИДЕОКАМЕРЫ, ОБРАБОТКА ВИДЕОПОТОКА, ПРИМЕНЕНИЕ СТАНДАРТОВ СЖАТИЯ ВИДЕОПОТОКОВ, ПРИМЕНЕНИЕ РАЗРАБОТАННОГО УСТРОЙСТВА

Цель проектирования: проектирование конструкции мобильной *IP*-видеокамеры.

Методология проведения работы: в процессе решения поставленных задач использованы методы проектирования электронных устройств, методы передачи и обработки видеопотока.

Результаты работы: спроектирована мобильная *IP*-видеокамера, проведен анализ передачи и обработки видеопотока, разработано программное обеспечение для обработки видеопотока.

Область применения результатов: разработанное устройство возможно применить для создания систем видеонаблюдения, которое может быть использовано в целях организации охраны периметра как в домашних условиях, так и на предприятии.

**СОДЕРЖАНИЕ**

## ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

| | | |
|---|---|---|
| САПР | – | Система автоматизированного проектирования |
| *AD* | – | *Altium Designer* |
| ЦП | – | Центральный процессор |
| *LWIP* | – | *Light weight internet protocol* |
| *GPIO* | – | *General-purpose Input/Output* |
| *API* | – | *Application programming interface* |
| *MII* | – | *Media independent onterface* |
| *RMII* | – | *Reduced media independent interface* |
| *DCMI* | – | *Digital camera interface* |
| *DMA* | – | *Direct memory access* |
| *DOM* | – | *Document object model* |
| *DFT* | – | Дискретное преобразование Фурье |
| *JS* | – | *JavaScript* |
| *PCB* | – | Печатная плата |
| *SMD* | – | Технология поверхностного монтажа |
| *PHY* | – | Физический уровень |
| *MAC* | – | *Media access control* |
| *SoC* | – | *System on Chip* |

# ВВЕДЕНИЕ

*IP*-видеокамера – это тип цифровой видеокамеры, которая получает и отправляет данные через сеть (*TCP/IP*). Такие камеры часто используются для наблюдения. В отличие от аналоговых камер видеонаблюдения (*CCTV*), для *IP*-камер не требуется локальное записывающее устройство, а требуется только локальная сеть. *IP*-камеры подключаются к сети так же, как и любое другое современное устройство, будь то смартфон или ноутбук.

Видеонаблюдение на сегодняшний день актуально как никогда. Системы видеонаблюдения выступают в качестве средства сдерживания преступности, помогают полиции в последующем расследовании. Системы видеонаблюдения защищают посетителей объекта как прямо, так и косвенно, каждого посетителя, который входит в здание, чтобы вести учёт подозрительной активности. Менеджеры компаний могут использовать видеонаблюдение для мониторинга производительности сотрудников, определения областей производительности труда, в которых сотрудник нуждается в улучшении, и обеспечения соблюдения сотрудниками правил безопасности компании. Работники по обслуживанию могут использовать камеры видеонаблюдения для обнаружения оборудования, требующего ремонта, и оборудования, работающего небезопасным образом.

Целью данной работы является проектирование, разработка конструкции портативной *IP*-видеокамеры. Основными компонентами устройства является микроконтроллер с ядром *ARM Cortex-M4*, а также модуль камеры *OV9655,* модуль ethernet, а также система питания устройства. Такое устройство должно уметь передавать видеопоток по локальной сети по стандарту *Ethernet*. В таком случае, к целям работы добавляется вопрос обработки и передачи видеопотока, а именно исследования методов сжатия на примере кодека *h.264* в контексте локальных сетей.

# 1 ОБЩЕТЕХНИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРИБОРА

## 1.1 Анализ исходных данных

Проектируемым прибором является мобильная *IP*-видеокамера. В качестве самой видеокамеры будет использоваться *OV9655 CameraBoard* – простая цифровая камера, основанная на *OV9655*, которая может как делать фотографии, так и снимать видео, но только при использовании совместно с отладочной платой. В основе такой платы будет микроконтроллер на базе ядра *ARM Cortex-M4*. Прибор должен уметь считывать, сжимать, передавать поток данных по *Ethernet*-каналу на веб-сервер. Основным источником питания должно быть питание от аккумулятора напряжением 3.6 В. Потребляемый ток должен быть не более 40 мА. Также должна быть предусмотрена возможность зарядки аккумулятора от дополнительного источника питания через разъём *RJ*-45 (технология *PoE*).

## 1.2 Теоретические сведения и принципы функционирования отдельных узлов прибора

### **1.2.1** Понятие изображения

Изображение – это визуальное представление чего-либо. В информационных технологиях этот термин имеет несколько значений. В основном, изображение – это картинка, созданная или скопированная и сохраненная в электронном виде. Изображение может быть описано с точки зрения векторной графики или растровой графики. Изображение, хранящееся в растровом виде, иногда называют растровым.

Векторная графика также известна как масштабируемая векторная графика (*SVG*). Эта графика состоит из закрепленных точек, соединенных линиями и кривыми. Поскольку эта графика не основана на пикселях, она известна как независимая от разрешения, что делает ее бесконечно масштабируемой. Линии в векторной графике четкие, без потери качества или детализации, независимо от их размера. Эта графика также не зависит от устройства, что означает, что ее качество не зависит от количества точек, доступных на принтере, или количества пикселей на экране. Поскольку они состоят из линий и опорных точек, размер файлов относительно невелик.

Растровые изображения состоят из пикселей или крошечных точек, которые используют цвет и тон для создания изображения. Пиксели выглядят как маленькие квадраты на миллиметровой бумаге, когда изображение увеличивается или увеличивается. Эти изображения создаются цифровыми камерами, путем сканирования изображений в компьютер или с помощью

растрового программного обеспечения. Каждое изображение может содержать только фиксированное количество пикселей; количество пикселей определяет качество изображения. Это известно как разрешение. Чем больше пикселей, тем лучше качество при том же или большем размере оригинала, но это также увеличивает размер файла и количество места, которое требуется для хранения файла. Чем меньше количество пикселей, тем ниже разрешение. Разрешение ограничивает размер изображения, которое может быть увеличено без возможности видеть пиксели. Однако изображение с высоким разрешением, напечатанное в маленьком размере, приведет к тому, что пиксели «втиснуты» друг в друга.

Общие форматы файлов изображений в Интернете включают [1]:

1 *JPEG* – это файл графического изображения, созданный в соответствии со стандартом Объединенной группы экспертов по фотографии, группы экспертов *ISO/IEC*, которая разрабатывает и поддерживает стандарты для набора алгоритмов сжатия файлов компьютерных изображений. Файлы JPEG обычно имеют расширение *.jpg*.

2 *GIF* означает формат обмена графикой. *GIF* использует растровый тип данных *2D* и закодирован в двоичном формате. Файлы *GIF* обычно имеют расширение *.gif*

3 *GIF89a* – это анимированное изображение в формате *GIF*, отформатированное в соответствии с версией *GIF 89a*. Одним из главных преимуществ формата является возможность создания анимированного изображения, которое можно воспроизвести после передачи на страницу просмотра, которая движется – например, вращающаяся иконка или баннер с машущей рукой или буквы, которые волшебным образом увеличиваются. *GIF89a* также может быть указан для чересстрочного представления *GIF*.

4 *PNG – Portable Network Graphics*) – это формат файлов для сжатия изображений, который был разработан для обеспечения ряда улучшений по сравнению с форматом *GIF*. Как и *GIF*, файл *PNG* сжимается без потерь (это означает, что вся информация об изображении восстанавливается при распаковке файла во время просмотра).

5 *SVG* – это масштабируемая векторная графика, описание изображения как приложения *XML*. Любая программа, например браузер, распознающая *XML*, может отобразить изображение, используя информацию, предоставленную в формате *SVG*. Масштабируемость означает, что файл можно просматривать на дисплее компьютера любого размера и разрешения, будь то маленький экран смартфона или большой широкоформатный дисплей ПК. Файлы обычно имеют расширение *.svg*

6 *TIFF* (*Tag Image File Format*) – это распространенный формат для обмена растровыми графическими (растровыми) изображениями между прикладными программами, включая те, которые используются для

изображений сканера. Файл *TIFF* можно идентифицировать как файл с суффиксом имени файла *.tiff* или «*.tif*».

7 *BMP* – формат несжатого растрового изображения, в заголовке которого записана информация об изображении – размер файла, ширина и высота рисунка, глубина пикселей, количество цветов. После заголовка может следовать палитра. Далее идет непосредственно набор данных о пикселях, который идентифицирует положение каждого пикселя и его цвет.

### **1.2.2** Принципы формирования и представления изображений

Как было отмечено ранее, векторный подход представляет изображение как совокупность простых элементов: прямых линий, дуг, окружностей, эллипсов, прямоугольников и других, которые называются графическими примитивами. Кодирование векторного изображения предполагает однозначное определение (задание) положения, формы, а также цветовых характеристик всех графических примитивов, составляющих рисунок [2].

Что касается растровой графики, то основным элементом растрового изображения является пиксель. Этот термин имеет несколько значений: отдельный элемент растрового изображения, отдельная точка на экране монитора, отдельная точка на изображении, напечатанном принтером. Поэтому на практике эти понятия часто обозначают так:

– пиксель – отдельный элемент растрового изображения;

– видеопиксель – элемент изображения на экране монитора;

– точка – отдельная точка, создаваемая принтером.

Растровое изображение состоит из пикселей. Каждый пиксель растрового изображения характеризуется координатами *x, y* и яркостью (только для черно-белых изображений). Поскольку пиксели имеют дискретный характер, то их координаты – это дискретные величины – целые или рациональные числа. В цветном изображении каждый пиксель характеризуется координатами x и y, и тремя яркостями: яркостью красного, яркостью синего и яркостью зеленого цветов (*VR, VB, VG*). Комбинируя данные три цвета, можно получить большое количество различных оттенков.

Если для представления каждого пикселя в черно-белом рисунке достаточно одного бита (бинарная форма записи), то для работы с цветом или полутоновым изображения этого явно недостаточно. Однако подход при кодировании цветных изображений остается неизменным. Любой рисунок разбивается на пиксели, т. е. небольшие части, каждая из которых имеет свой цвет.

Объем информации, описывающий цвет пикселя, определяет глубину цвета. Чем больше информации определяет цвет каждой точки в рисунке, тем больше вариантов цвета существует. Понятно, что для рисунков в естественном цвете требуется больший объем памяти. Чтобы представить

более шестнадцати миллионов цветов, информация о каждой точке рисунка должна занимать четыре байта, что в тридцать два раза больше, чем для монохромного рисунка.

Важной характеристикой изображения является разрешение, которое измеряется в количестве точек на дюйм.

Разбив рисунок на пиксели, описав цвет каждого пикселя и задав разрешение, можно закодировать любое изображение. Таким образом, программы могут воспроизводить изображения.

### **1.2.3** Классификация цветовых моделей

Цветовая модель – термин, обозначающий абстрактную модель описания представления цветов в виде кортежей чисел, обычно из трёх или четырёх значений, называемых цветовыми компонентами или цветовыми координатами. Вместе с методом интерпретации этих данных множество цветов цветовой модели определяет цветовое пространство.

Все цветовые модели можно разделить на три группы, в зависимости от аппаратной платформы [3]:

– аппаратно- зависимые модели, используемые в технических средствах ввода и вывода графической информации (*RGB, CMYK*);

– аппаратно-независимые модели, описывающие цвет в абстрактных колориметрических терминах (*XYZ*);

– интуитивные модели, построенные на основе субъективного восприятия цвета человеком (*HSB*).

Также в зависимости от объекта, на котором воспроизводится изображение, цветовые модели можно поделить на аддитивные и субтрактивные.

Модель *RGB* используется при работе с проектами на основе цифрового экрана, например, при просмотре на экране компьютера или дисплее телефона. В цветовой модели *RGB* каждому из основных цветов, красному, зеленому и синему, присваивается значение от 0 до 255, где 0 – темный, а 255 – яркий. Указав три значения для красного, зеленого и синего цветов, можно указать точный цвет, который получится в итоге.

Цветовая модель *RGB* представляет собой аддитивную цветовую систему, что означает, что цвета при смешивании становятся светлее. Когда каждый компонент света смешивается, комбинация становится новым цветом. Красный, зеленый и синий – это три аддитивных основных цвета.

Экраны телевизоров и компьютерные мониторы создают цвет, включая красный, зеленый и синий основные цвета в каждом пикселе.

Поскольку цветовая модель *RGB* способна воспроизводить только определенный диапазон (или гамму) цветов, некоторые цвета не могут быть точно воспроизведены монитором компьютера. Количество цветов, видимых

на мониторе, еще больше уменьшается из-за ограничений видеооборудования компьютера, которое может отображать любое изображение, от черно-белого до 16,7 миллионов цветов.

Цветовая модель *CMYK* описывает цвета на основе процентного содержания голубого, пурпурного, желтого и черного. Многие компьютерные принтеры и традиционные «четырехцветные» печатные машины используют модель *CMYK*. В модели *CMYK*, смешивая голубые, пурпурные, желтые и черные чернила или краски, можно создать практически любой желаемый цвет.

*CMYK* – это субтрактивная цветовая модель, означающая, что при смешивании цвета становятся темнее. Каждая из смешанных красок или чернил поглощает различные компоненты света. Если смешать правильную комбинацию красок, все компоненты света поглощаются, и в результате получается почти черный цвет. Однако в реальном мире чернила, которые используют принтеры, не идеальны поэтому в результате смешивания цветов получается грязно-коричневый цвет, потому что основные цвета перекрываются и не полностью вычитают свет при смешивании. Поэтому букву *K* в аббревиатуре обозначает слово *Key* или чёрный цвет.

Модель *HSL* очень похожа на цветовую модель *RGB*. На самом деле, когда они выражены математически, они идентичны. Разница заключается в том, как цвета выражаются численно.

Оттенок определяет, какой это основной цвет. Красный, зеленый, синий, желтый, оранжевый и т. д. – это разные оттенки. Насыщенность и яркость больше говорят о вариациях этих основных цветов. Насыщенность – это яркость (или «чистота») цвета, т. е. то, сколько дополнительных цветов смешано. Наконец, легкость относится к «белизне» цвета. Его также можно назвать «яркостью», «значением» или «интенсивностью».

Другими моделями, связанными с моделью *HSL*, являются модели *HSB* (оттенок, насыщенность, яркость) и *HSI* (оттенок, насыщенность, интенсивность). Все эти термины похожи, но не взаимозаменяемы.

### **1.2.4** Разновидности цветовой модели *RGB*

Существует много разновидностей цветовой модели *RGB*, все они отличаются количеством бит, отведённых под каждый цвет. Например, в *9-bit RGB* на каждый цвет отведено по 3 бита, значит цветовая палитра включает в себя 512 цветов.

Наиболее используемым форматом является *RGB24* [4], где каждому цвету соответствует один байт. Таким образом, *RGB24* может передавать почти 17 миллионов цветов. Как можно заметить, каждый такой формат отличается определённым параметром, который напрямую влияет на

количество передаваемых цветов и который называется глубиной цвета (рисунок 1).

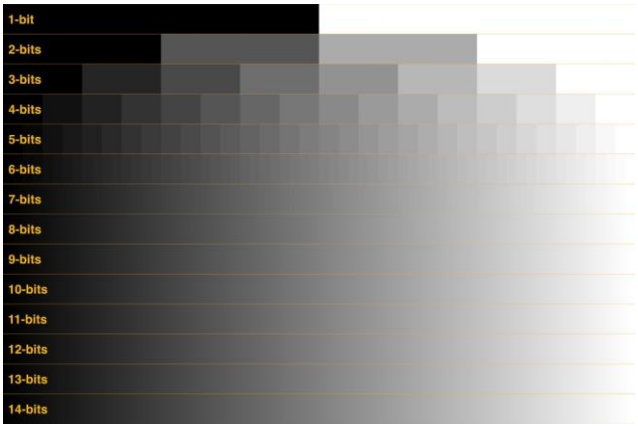



Рисунок 1 – Глубина цвета на примере черного и белого цвета

Также существует *RGB32*, который отличается от *RGB24* тем, что к нему добавляется ещё 1 байт под так называемый альфа-канал, который отвечает за прозрачность пикселя.

Битовая глубина (или глубина цвета) определяет количество возможных значений цвета, но также существует цветовое пространство, которое определяет максимальное значение или диапазон цвета. Что касается цветового пространства, то кроме стандартного *RGB* есть его модификации, такие как *Rec.709, AdobeRGB, DCI-P3, Rec.2020*. Как можно заметить на рисунке 2, *Rec*.2020, например, имеет большой охват зелёного цвета, соответственно, он может передать больше цветов.

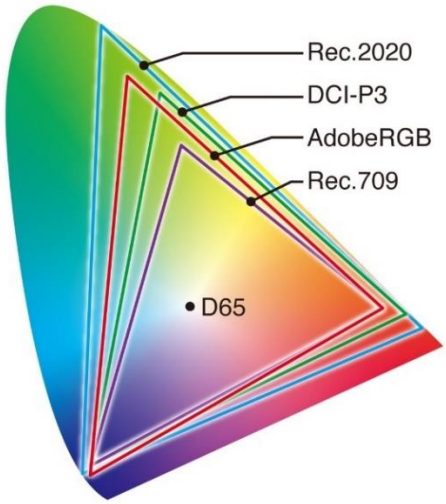



Рисунок 2 – Разница цветовых диапазонов

*SRGB* по-прежнему является стандартом для компьютерных изображений, большинства потребительских и средних фотокамер и домашних принтеров. Для профессиональной печати и допечатной подготовки часто используется *Adobe RGB* с расширенным цветовым охватом, который можно воспроизвести с помощью профессиональной печати *CMYK*.

### **1.2.5** Понятие видеопотока

Потоковое видео – это технология, которая позволяет пользователю просматривать онлайн-видеоконтент без необходимости предварительной загрузки медиа файлов. Потоковое видео относится конкретно к онлайн-видеоконтенту, такому как фильмы, телешоу, прямые трансляции событий и онлайн-видео, видеонаблюдение.

Поскольку пользователю не нужно загружать медиа файлы для использования контента, потоковая передача помогает экономить ресурсы хранения, время, затрачиваемое на загрузку или буферизацию видео, и обеспечивает удобство просмотра для пользователей. Потоковая передача видео (аудио) не была бы возможна без специальных протоколов.

Протоколы потоковой передачи видео [5] – это специальные стандартизированные правила и методы, которые разбивают видеофайлы на более мелкие части (пакеты), чтобы их можно было доставить конечному пользователю для повторной сборки и просмотра

Файлы должны быть сжаты для транспортировки, этот процесс достигается с помощью «кодека», такого как наиболее распространенный *H.264*. Файлы также должны быть сохранены в «контейнерном формате», таком как *.mp4* или *.avi*, прежде чем их можно будет передать. Источником видеофайла может быть непосредственно камера вещателя в случае прямой трансляции или статические файлы в случае видео по запросу (*VoD*).

На сегодняшний день существует несколько протоколов потоковой передачи видео. Некоторые из них являются устаревшими стандартами, которые все еще используются в некоторых случаях, в то время как другие быстро развиваются, особенно благодаря среде с открытым исходным кодом.

На сегодняшний день *HLS* является наиболее часто используемым протоколом для потоковой передачи Этот протокол совместим с широким спектром устройств, от настольных браузеров, смарт-телевизоров, телевизионных приставок, мобильных устройств *Android* и *iOS* и до видеоплееров *HTML5*.

*HLS* также поддерживает потоковую передачу с адаптивным битрейтом. Единственным серьезным недостатком, связанным с протоколом *HLS*, могут быть связанные с ним большие задержки. Под задержкой понимается время, необходимое информации для перемещения от источника к месту назначения и обратно, когда через Интернет передаются большие объемы данных.

*MPEG-DASH* – один из последних протоколов потоковой передачи, разработанный Экспертной группой по движущимся изображениям (*MPEG*) в качестве альтернативы стандарту *HLS*. Это стандарт с открытым исходным кодом, который можно настроить для любого аудио- или видеокодека.

Как и *HLS, MPEG-DASH* поддерживает потоковую передачу с адаптивным битрейтом, позволяя зрителям получать видео самого высокого качества, которое может обрабатывать их сеть.

*WebRTC* – это проект с открытым исходным кодом, целью которого является доставка потоковой передачи с минимальной задержкой в реальном времени. Первоначально разработанный для приложений на основе исключительно чата и использования *VoIP*, он стал известен для использования в приложениях для видеочата и конференций после того, как был куплен *Google*.

*SRT* – это еще один протокол с открытым исходным кодом, разработанный поставщиком технологий потоковой передачи *Haivision*. Это предпочтительный протокол для членов *SRT Alliance*, группы компаний, в которую входят поставщики технологий и телекоммуникаций. Основными преимуществами, которыми славится *SRT*, являются безопасность, надежность, совместимость и потоковая передача с малой задержкой.

*SRT* может передавать потоковое видео высокого качества, даже если сетевые условия нестабильны. Он также не зависит от одного кодека, что позволяет использовать его с любыми аудио- и видеокодеками.

### **1.2.6** Назначение видеокамеры

Основное назначение цифровой видеокамеры [6] – преобразование оптического изображения в электрический сигнал. Чтобы это было возможно, камере необходимо иметь в себе как минимум три компонента: объектив, матрицу, плату обработки.

В то время как свет отражается от объектов, он также может проходить сквозь объекты, но, когда это происходит, он может менять направление. Объектив камеры улавливает все световые лучи, отражающиеся вокруг, и использует стекло, чтобы перенаправить их в одну точку, создавая четкое изображение

Когда все эти световые лучи встречаются на сенсоре цифровой камеры или на куске пленки, они создают четкое изображение. Если свет не встречается в нужной точке, изображение будет выглядеть размытым или не в фокусе. Система фокусировки объектива перемещает линзу ближе или дальше от сенсора или пленки, позволяя фотографу настроить объектив так, чтобы объект был резким.

Объектив проецирует свет, который отражается от объектов, на матрицу. Качество детализации объекта зависит от расстояния до него. Чем

большая степень детализации требуется, тем большую часть матрицы должно занимать интересующее нас изображение.

Регулируется это подбором фокусного расстояния объектива или углом его обзора. Эти два параметра находятся в непосредственной взаимосвязи – чем больше фокусное расстояние, тем меньше угол обзора.

Матрицей называется пластина из полупроводникового материала, которая состоит из множества светочувствительных участков (пикселей). Чем больше пикселей, тем лучше разрешение камеры и, соответственно, детализация. Однако при неизменной площади пластины увеличение количества пикселей приводит к потере светочувствительности камеры.

Плата обработки сигнала формирует выходной сигнал и в зависимости от вида этого сигнала (аналоговый и цифровой) существуют аналоговые и цифровые камеры.

Аналоговые видеокамеры умеют только передавать сигнал, их функциональные возможности позволяют этот сигнал обрабатывать с целью улучшения и устранения возможных помех, искажений.

Цифровые видеокамеры также называются *IP*-видеокамерами или сетевыми видеокамерами. В отличие от аналоговых, сетевые видеокамеры могут работать в локальных сетях как отдельные устройства. Также сетевые видеокамеры способны как записывать видео на встроенные карты памяти, так и передавать видеопоток на сервер.

### **1.2.7** Ethernet в контексте модели *OSI*

Модель *OSI* – эталонная модель, которая описывает архитектуру и принципы работы компьютерных сетей. Всего данная модель имеет 7 уровней. Так как речь идёт про *Ethernet*, то будут рассмотрены два первых уровня в данной сетевой модели – физический и канальный.

Физический уровень в *OSI* отвечает за передачу непосредственно электрических сигналов (битов) по какой-либо среде и также представляет собой физическую среду, по которой передаются сигналы между узлами. Формат таких сигналов варьируется в зависимости от среды передачи. В случае *Ethernet* биты передаются в виде электрических импульсов. В случае *Wi-Fi* биты передаются в виде радиоволн. В случае оптоволокна биты передаются в виде световых импульсов.

Второй уровень называет канальным, и он решает проблему адресации при передаче информации. Канальный уровень получает биты и превращает их в кадры – *frames*. Канальный уровень формирует кадры с адресом отправителя и получателя, после чего отправляет их по сети. Кадр – это единица данных, которыми обмениваются компьютеры в сети *Ethernet*. Кадр имеет фиксированный формат и наряду с полем данных содержит различную служебную информацию, например адрес получателя и адрес отправителя.

После того как адаптер отправителя поместил кадр в сеть, его начинают принимать все сетевые адаптеры. Каждый адаптер проводит анализ кадра, и если адрес совпадает с их собственным адресом устройства (*MAC*–адрес), кадр помещается во внутренний буфер сетевого адаптера, если же не совпадает, то он игнорируется.

У канального уровня есть два подуровня – это *MAC* и *LLC*. *MAC* расшифровывается как *Media Access Control* и отвечает за присвоение физических *MAC*-адресов, а *LLC – Logical Link Control –* отвечает за контроль логической связи и занимается проверкой и исправлением данных, управляет их передачей.

Ethernet работает на первых двух уровнях модели *OSI* [7]. На втором уровне *Ethernet* разделяет функции канального уровня передачи данных на два отдельных подуровня: подуровень управления логическим каналом (*LLC*) и подуровень управления доступом к среде (*MAC*). Использование этих подуровней значительно способствует совместимости между различными конечными устройствами. Подуровень *Ethernet MAC* выполняет две основные функции: инкапсуляция данных и контроль доступа к данным. *MAC* включает в себя инициирование передачи кадров и восстановление после сбоя передачи из–за коллизий. *LLC* управляет связью между верхними уровнями и сетевым программным обеспечением, а также нижними уровнями, как правило, аппаратным обеспечением.

Основной принцип работы, используемый в данной технологии, заключается в следующем: для того чтобы начать передачу данных в сети, сетевой адаптер компьютера «прослушивает» сеть на наличие какого–либо сигнала, и если его нет, то адаптер начинает передачу данных, если же сигнал есть, то передача откладывается на определенный интервал времени. Время монопольного использования разделяемой среды одним узлом ограничивается временем передачи одного кадра.

### **1.2.8** Протокол *TCP/IP*

Интернет-протокол (*IP*) – это адресная система Интернета, основная функция которой заключается в доставке пакетов информации от исходного устройства к целевому устройству. *IP* – это основной способ установления сетевых соединений, и он закладывает основу компьютерных сетей. *IP* не обрабатывает порядок пакетов или проверку ошибок. Для такой функциональности требуется другой протокол, обычно *TCP*.

*IP* – это протокол без установления соединения, что означает, что каждая единица данных адресуется и маршрутизируется от исходного устройства к целевому устройству индивидуально, а целевое устройство не отправляет подтверждение обратно источнику. Вот тут появляются такие протоколы, как протокол управления передачей (*TCP*). *TCP* используется в сочетании с *IP* для

поддержания соединения между отправителем и получателем и для обеспечения порядка пакетов.

*TCP/IP* определяет, как устройства передают данные между собой, причём важно, чтобы эти данные никак не изменялись во время своего пути [8]. Чтобы гарантировать, что каждое сообщение достигает адресата без изменений, модель *TCP/IP* разбивает данные на пакеты, а затем повторно собирает пакеты в полное сообщение на другом конце. Отправка данных небольшими пакетами упрощает поддержание точности по сравнению с отправкой всех данных сразу.

После разделения одного сообщения на пакеты эти пакеты могут перемещаться по разным маршрутам, если один маршрут перегружен. Это как отправить по почте несколько разных поздравительных открыток одному и тому же дому.

Таким образом, *TCP/IP* – сетевая модель, которая описывает процесс передачи цифровых данных, и которая была названа в честь двух главных протоколов в себе на время создания [9]. Как и модель *OSI*, данная модель имеет уровни, но их 4, а не 7: прикладной, транспортный, сетевой и канальный. На каждом уровне работают свои протоколы (рисунок 3), поэтому *TCP/IP* является стеком протоколов.

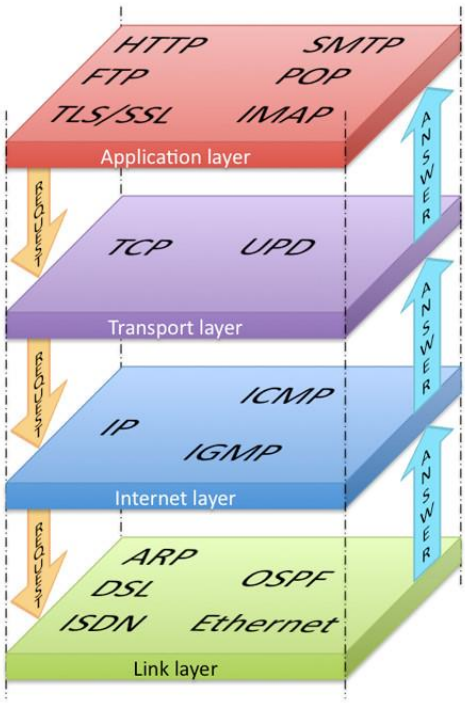



Рисунок 3 – Модель *TCP/IP*

Канальный уровень обрабатывает физический обмен электрическими сигналами. На канальном уровне, как правило, работает протокол *Ethernet*. По

этому стандарту каждое устройство имеет свой уникальный *MAC*-адрес, который затем инкапсулируется в пакет данных на сетевом уровне.

Сетевой уровень формирует пакеты данных, добавляет *IP*-адреса источника и получателя к заголовкам пакетов.

Далее данные инкапсулируются на транспортный уровень, где пакеты становятся сегментами (если речь идёт про *TCP*) или дейтаграммами (*UDP*). Выделяются порядковые номера, в заголовок добавляются номера портов источника и получателя. В случае с *TCP*, который является протоколом гарантированной доставки, доставка подтверждается, а то, что потерялось, отсылается повторно.

Уровень приложения использует протоколы самого верхнего уровня, чтобы каким-либо образом отображать, изменять данные в понятном человеку виде. Схема инкапсуляции, декапсуляции, а также содержимого пакетов представлено на рисунке 4.

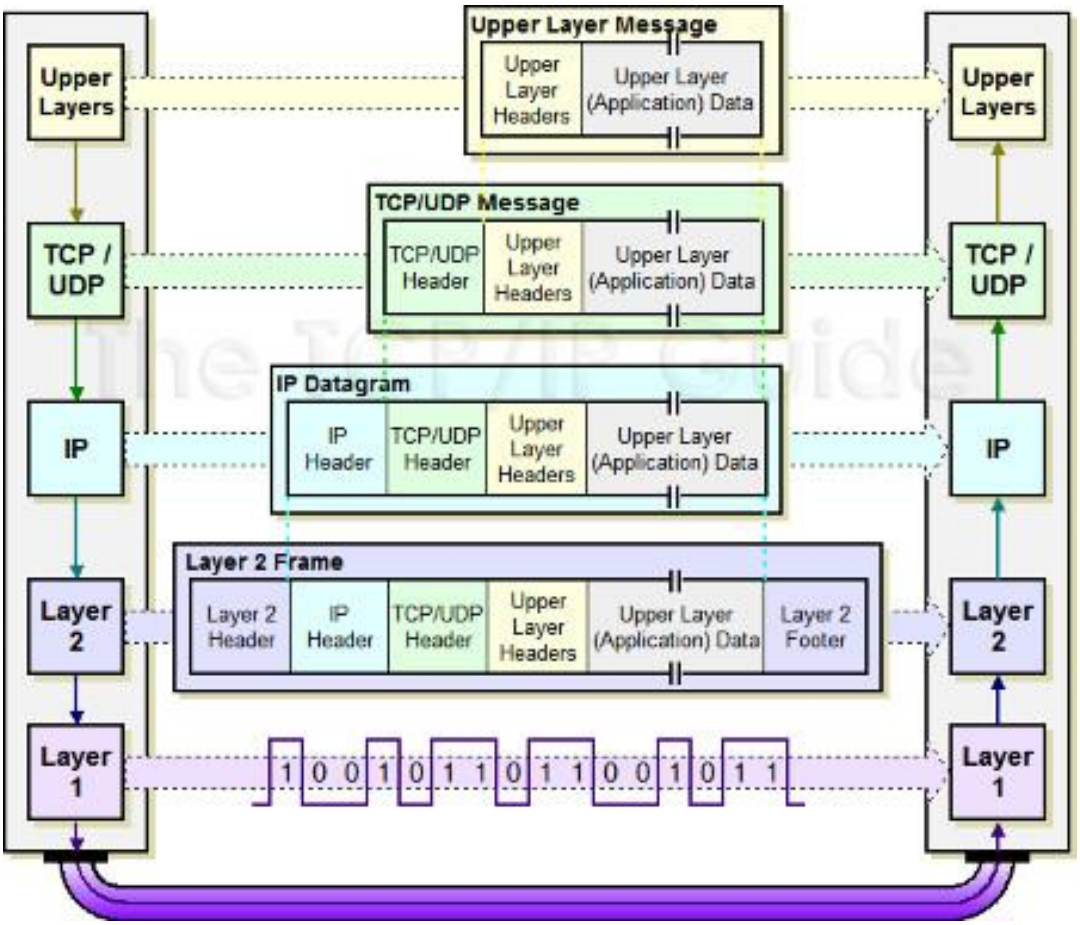



Рисунок 4 – Обработка пакетов в *TCP/IP*

Таким образом, модель *TCP/IP* остается неизменной, в то время как стандарты протоколов могут обновляться, что еще дальше упрощает работу с *TCP/IP*. Благодаря всем преимуществам стек *TCP/IP* получил широкое

распространение и использовался сначала в качестве основы для создания глобальной сети, а после для описания работы интернета.

### **1.2.9** Обзор современных архитектур и микропроцессорной базы *IP*-видеокамер

В 1974 году компания *Texas Instruments* выпустила первый микроконтроллер, представляющий собой тип микропроцессора с оперативной памятью и устройствами ввода-вывода, интегрированными с ЦП на одном кристалле. Вместо того, чтобы соединять отдельные компоненты, их объединили на одном чипе.

В последнее время развитие мобильных систем подтолкнуло интеграцию даже дальше, чем микропроцессоры или микроконтроллеры. В результате получается система на чипе, которая может упаковать многие элементы современной компьютерной системы (графический процессор, сотовый модем, ускорители ИИ, *USB*-контроллер, сетевой интерфейс) вместе с процессором и системной памятью в одном корпусе. Это еще один шаг в непрерывной интеграции и миниатюризации электроники, которая, вероятно, будет продолжаться и в будущем.

*SoC* – это аббревиатура от *System on Chip* [10], это основа каждой *IP*-камеры (а также *DVR* и *NVR*). Для *IP*-камер *SoC* объединяет процессор сигналов изображения (*ISP*), модуль видеокодека (кодирование), процессор цифровых сигналов (*DSP*), сетевые периферийные интерфейсы, интеллектуальный механизм анализа видео и т. д.

Если знать, какая *SoC* использовалась, можно сравнивать *IP*-камеры разных брендов, которые имеют схожие функции. Однако многие производители *IP*-камер не раскрывают *SoC*, который они используют в своих продуктах. *Amabrella, Hisilicon, Texas Instruments* – три известных и распространенных производителя *SoC* для *IP*-камер [11]. Известные бренды оборудования для *IP*-наблюдения, такие как *Hikvision* и *Dahua*, в основном *используют SoC Ambarell*а, а *Uniview* и *Tiandy* используют *SoC Hisilicon*.

Сегодня многие *SoC* основаны на процессорных ядрах *Arm*, таких как ядра *Cortex-A, Cortex-M* и *Cortex-R*. Другие специализированные ядра, используемые в архитектурах *SoC*, включают процессоры *Synopsys ARC*, процессоры *Cadence Tensilica Xtensa* и процессорные ядра, основанные на архитектуре набора инструкций *RISC-V* [12].

Архитектуры *SoC* все чаще основаны на нескольких ядрах. При симметричной многопроцессорной обработке вычисления распределяются по нескольким процессорным ядрам. В асимметричной многопроцессорной обработке у ядер могут быть совершенно разные роли: одни выполняют задачи управления вводом-выводом в реальном времени, а другие выполняют исполнительные функции.

Архитектуры *SoC* могут включать в себя различные типы памяти и конфигурации. Статическая оперативная память (*SRAM*) может использоваться для регистров процессора и быстрых кэшей уровня 1 (или *L1*), в то время как динамическая оперативная память (*DRAM*) часто составляет основную память нижнего уровня *SoC*.

Многие периферийные устройства были включены в архитектуры *SoC*, часто для работы с часто используемыми протоколами связи. Популярные интерфейсы включают *GPIO, PCI-Express, Gigabit Ethernet, CAN, SPI, USB, UART* и *I2C*.

### **1.2.10** Структура микроконтроллерного ядра *ARM Cortex–M4*

Аббревиатура *ARM* известна как *Acorn RISC Machines*, но со временем она была переименована на *Advanced RISC Machines*. Важным моментом здесь является то, что *ARM* не разрабатывает кремниевые чипы для микроконтроллеров, а только предоставляет ядро для микропроцессора и других строительных блоков микроконтроллера. Затем *ARM* предоставляет его различным производителям посредством лицензирования.

Микроконтроллеры *Cortex-M* представляют собой одно из направлений развития микропроцессорных ядер, предлагаемых фирмой *ARM*. Фактически, под общей торговой маркой *Cortex* можно увидеть три типа процессоров, обозначаемых буквами *A, R, M*. Профиль *A* характеризуется достижением большой вычислительной мощности. Профиль *R* нацелен на использование во встраиваемых системах, поэтому эти процессоры модернизированы для исполнения задач в реальном времени. Основной задачей профиля *M* заявлена простота и низкая стоимость. Технически *Cortex-M* представляют сильно упрощенные варианты старших моделей. Тем не менее, даже такие «урезанные» контроллеры обладают вычислительной мощностью, значительно превышающей многие аналоги.

Семейство *Cortex-M* состоит из *Cortex-M0, Cortex-M0+, Cortex-M1, Cortex-M3, Cortex-M4 и Cortex-M7*. Рассматриваемое ядро *Cortex-M4* основано на архитектуре *ARMv7*.

Структура *ARM Cortex-M4* представлена рисунке 5.

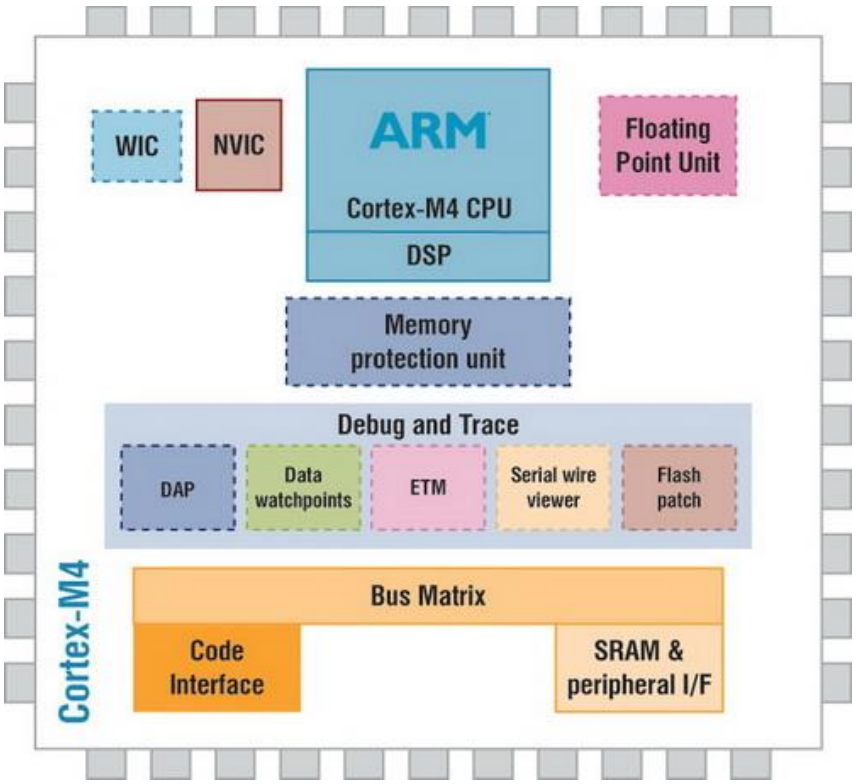



Рисунок 5 – Структура *ARM Cortex-M4*

*ARM Cortex-M4* состоит из следующих компонентов [13]:
– ядро процессора;
– контроллер вложенных векторных прерываний;
– блок с системой отладки;
– система шин и матрица шин;
– память;
– блок операций с плавающей точкой;

Микроконтроллеры с ядром *ARM Cortex-M4* поддерживают 240 системных и периферийных прерываний, а *NVIC* осуществляет управление прерываниями с помощью таблицы векторов прерываний. Также микроконтроллеры *ARM Cortex-M4* обычно основаны на 32-разрядной архитектуре *RISC*.

### **1.2.11** Регистровая модель портов ввода–вывода общего назначения микроконтроллера с ядром *ARM Cortex–M4*

*GPIO* – это сигнальный контакт на интегральной схеме или плате, который можно использовать для выполнения функций цифрового ввода или вывода [14]. По своей конструкции он не имеет заранее определенной формы и может использоваться разработчиком аппаратного или программного обеспечения для выполнения выбранных им функций. Типичные области

применения включают в себя управление светодиодами, переключателями считывания и управление различными типами датчиков.

*GPIO* может быть реализован с помощью выделенных интегральных схем или поддерживаться устройствами системы на кристалле (*SoC*) или системой на модуле (*SoM*).

Что касается *ARM Cortex-M4* и *GPIO*, то важно упомянуть про разделение пространства памяти на блоки, часть которых представлена на рисунке 6.

| | |
|---|---|
| 0x4002 2C00 - 0x4002 2FFF | Reserved |
| 0x4002 2800 - 0x4002 2BFF | |
| 0x4002 2400 - 0x4002 27FF | |
| 0x4002 2000 - 0x4002 23FF | |
| 0x4002 1C00 - 0x4002 1FFF | GPIOH |
| 0x4002 1800 - 0x4002 1BFF | GPIOG |
| 0x4002 1400 - 0x4002 17FF | GPIOF |
| 0x4002 1000 - 0x4002 13FF | GPIOE |
| 0X4002 0C00 - 0x4002 0FFF | GPIOD |
| 0x4002 0800 - 0x4002 0BFF | GPIOC |
| 0x4002 0400 - 0x4002 07FF | GPIOB |
| 0x4002 0000 - 0x4002 03FF | GPIOA |

Рисунок 6 – Разметка участка памяти

Как можно увидеть, *GPIO* разделён на *GPIO A, GPIO B, GPIO C, GPIO D, GPIO E, GPIO F, GPIO G, GPIO H*.

Каждый порт в контроллерах с ядром *ARM Cortex-M4* состоит из 16 контактов [15]. С каждым выводом связаны различные регистры, изменяя содержимое регистров, можно управлять поведением конкретного вывода.

В контроллерах с ядром *ARM Cortex-M4* поведение каждого вывода можно контролировать с помощью:

– *GPIO Mode Register* (регистр режима *GPIO*);

– *GPIO Output Type Register* (регистр типа вывода *GPIO*);

– *GPIO Speed Registe*r (регистр скорости *GPIO*);

– *GPIO Pull-up/Pull-down Register* (подтягивающий/вытягивающий регистр *GPIO*);

– *GPIO Input data Register* (регистр входных данных *GPIO*);

– *GPIO Output data Register* (регистр выходных данных *GPIO*);

– *GPIO bit set/reset Register* (установка/сброс бита);

– *GPIO Configuration Lock Register* (регистр блокировки конфигурации *GPIO*);

– *GPIO Alternate Functionality Register* (регистр альтернативной функциональности *GPIO*).

Регистр режима *GPIO* используется для выбора режима вывода. В этом регистре можно запрограммировать четыре режима: режим ввода, режим вывода общего назначения, режим альтернативной функции и аналоговый режим. Режим вывода общего назначения используется, когда нужно что-то подать на выход. Каждый вывод имеет соответствующий выходной буфер, который может быть записан программным обеспечением. Альтернативный функциональный режим используется, когда нужно назначить конкретный вывод любому другому периферийному устройству, например: если используется *I2C,* то понадобятся два контакта, для *SDA* и *SCL*. Аналоговый режим работы если нужны возможности аналого-цифрового или цифро-аналогового преобразователя.

Регистр типа вывода *GPIO* – этот регистр записывается программным обеспечением для настройки типа вывода. Возможны два типа выхода: выход двухтактный и выход с открытым стоком.

Регистр скорости *GPIO* – это параметр скорости, который определяет, насколько быстро он может перейти от уровня 0 к уровню логической единицы и наоборот.

Регистр *GPIO Pull-up/Pull-down*. Используя этот регистр, можно включать/отключать управлять током проводящей линии на месте контакта.

Регистр входных данных *GPIO*. Этот регистр можно использовать для установки состояния чтения на выводе. Может считывать до 4 байтов.

Регистр выходных данных *GPIO* – этот регистр используется для установки значения 0 или 1 на выходном контакте.

Регистр установки/сброса битов *GPIO*. Этот регистр позволяет устанавливать и сбрасывать каждый отдельный бит в регистре выходных данных *GPIO*.

Регистр блокировки конфигурации *GPIO* – этот регистр используется для блокировки конфигурации вывода.

Регистр альтернативных функций *GPIO*. Этот регистр используется для настройки контактов в качестве альтернативных функций, т. е. эти контакты могут использоваться другими периферийными устройствами в микроконтроллере. Однако конкретный вывод может быть связан только с одним периферийным устройством одновременно.

### **1.2.12** Физический и канальный уровни интерфейса *I2C*

Протокол связи *I2*C (*Inter-Integrated Circuit*) был разработан компанией *Philps*. В настоящее время мало где используется ввиду низкой пропускной способности, но широко использовался для связи между несколькими

интегральными схемами (ИС) в системе. Состоит из двух уровней – физического и канального [16].

Физический уровень представлен шиной, в которой есть две линии шины: двунаправленная последовательная линия данных (*SDA*) и последовательная линия синхронизации (*SCL*). Линия данных используется для представления данных, а линия синхронизации используется для синхронизации передачи и приема данных.

Каждое устройство, подключенное к шине, имеет независимый адрес, и хост может использовать этот адрес для доступа между различными устройствами.

На шине *I2C* высокий уровень представляет собой логическую 1, низкий уровень представляет собой логический 0, а другое состояние является состоянием с высоким импедансом.

Когда несколько хостов используют шину одновременно, для предотвращения конфликтов данных используется арбитраж для определения того, какое устройство занимает шину.

*I2C* имеет три режима передачи: стандартный режим имеет скорость передачи 100 кбит/с, быстрый режим – 400 кбит/с, а высокоскоростной режим может достигать 3,4 Мбит/с, но большинство устройств *I2C* в настоящее время не поддерживают высокоскоростной режим.

Количество микросхем, подключенных к одной шине, ограничено максимальной емкостью шины 400 пФ.

Логический уровень *I2C* определяет сигналы запуска и остановки, достоверность данных, ответ, арбитраж, синхронизацию тактов и адресную широковещательную передачу связи.

После того, как сгенерирован стартовый сигнал, все ведомые устройства начинают ждать адресного сигнала ведомого устройства (*SLAVE_ADDRESS*), которое затем передает ведущее устройство. На шине *I2C* адрес каждого устройства уникален. Когда адрес, передаваемый хостом, совпадает с адресом определенного устройства, устройство выбирается, а невыбранное устройство игнорирует последующий сигнал данных. В соответствии с протоколом *I2C* этот адрес ведомого устройства может быть 7- или 10-битным.

После адресного бита идет бит выбора направления передачи. Когда этот бит равен 0, это означает, что последующее направление передачи данных – от хоста к ведомому, то есть хост записывает данные в ведомое устройство. Когда этот бит равен 1, происходит обратное, то есть ведущий считывает данные с ведомого.

После того, как ведомое устройство получит соответствующий адрес, ведущее или ведомое устройство вернет сигнал подтверждения (*ACK* = 0) или отклонения (*NACK* = 1).

Только после получения сигнала подтверждения хост может продолжать отправлять или получать данные.

Размер пакета данных составляет 8 бит, и хост будет отправлять его каждый раз. Для байтовых данных необходимо дождаться ответного сигнала (*ACK*) от ведомого устройства. Чтобы повторять неограниченное число байтов данных, нужно просто повторять этот процесс. Когда передача данных завершена, мастер отправляет подчиненному сигнал остановки передачи, что означает, что данные больше не передаются.

Когда ведущий хочет прекратить прием данных, он возвращает сигнал отклонения (*NACK*) ведомому, и мастер автоматически прекращает передачу данных.

Когда линия *SCL* имеет высокий уровень, линия *SDA* переключается с высокого уровня на низкий. Эта ситуация свидетельствует о начале общения.

Когда *SCL* находится на высоком уровне, линия *SDA* переключается с низкого уровня на высокий уровень, указывая на остановку связи.

*I2C* использует сигнальную линию *SDA* для передачи данных и сигнальную линию *SCL* для синхронизации данных. Линия данных *SDA* передает один бит данных в каждом такте *SCL*.

### **1.2.13** Регистровая модель *I2C* микроконтроллера на базе ядра *ARM Cortex–M4*

Регистровая модель *I2C* у *STM32* представлена на рисунке 7. Регистр управления *I2C_CR1* содержит в себе [17]:

– *SWRST(Software reset)* – единица в этом бите сбрасывает значение всех регистров модуля в базовое стояние, может использоваться для сброса при возникновении ошибки.

– *ALERT(SMBus alert)* – установка единицы в этот бит разрешает генерировать сигнал *alert* в режиме *SMBus*;

– *PEC(Packet error checking)* – управление этим битом производится программно, но он может быть сброшен аппаратно, когда передается *PEC, START, STOP* или *PE*=0. Единица в этом бите разрешает передачу *CRC*;

– *POS(Acknowledge/PEC Position (for data reception))* – состояние этого бита определяет положение *ACK/PEC* в двух байтовой конфигурации в режиме *Master*;

– *ACK(Acknowledge enable)* – единица в этом бите разрешает отправлять *ACK/NACK* после приема байта адреса или данных;

– *STOP(Stop generation)* – установка единицы в этот бит генерирует сигнал *STOP* в режиме *Master*;

– *START(Start generation)* – установка единицы в этот бит генерирует состояние *START* в режиме *Master*;

– *NOSTRETCH(Clock stretching disable (Slave mode))* – если на обработку данных требуется время *Slave* может остановить передачу мастера, прижав

линию *SCL* к земле, *Master* будет ждать и не будет ни чего слать, пока линия не будет отпущена. Ноль в этом бите прижимает SCL к земле;

– *ENGC(General call enable)* – если в этом бите установлена единица, модуль отвечает *ACK* на широковещательный адрес 0x00.

– *ENPEC(PEC enable)* – установка единицы в этот бит включает аппаратный подсчет *CRC*;

– *ENARP(ARP enable)* – установка единицы в этот бит включает *ARP*;

– *SMBTYPE(SMBus type)* – если в этом бите установлен ноль модуль работает в режиме *Slave*, если единица в режиме *Master*;

– *SMBUS(SMBus mode)* – если в этом бите установлен ноль модуль работает в режиме *I2C*, если единица *SMBus*;

– *PE(Peripheral enable)* – единица в этом бите включает модуль.

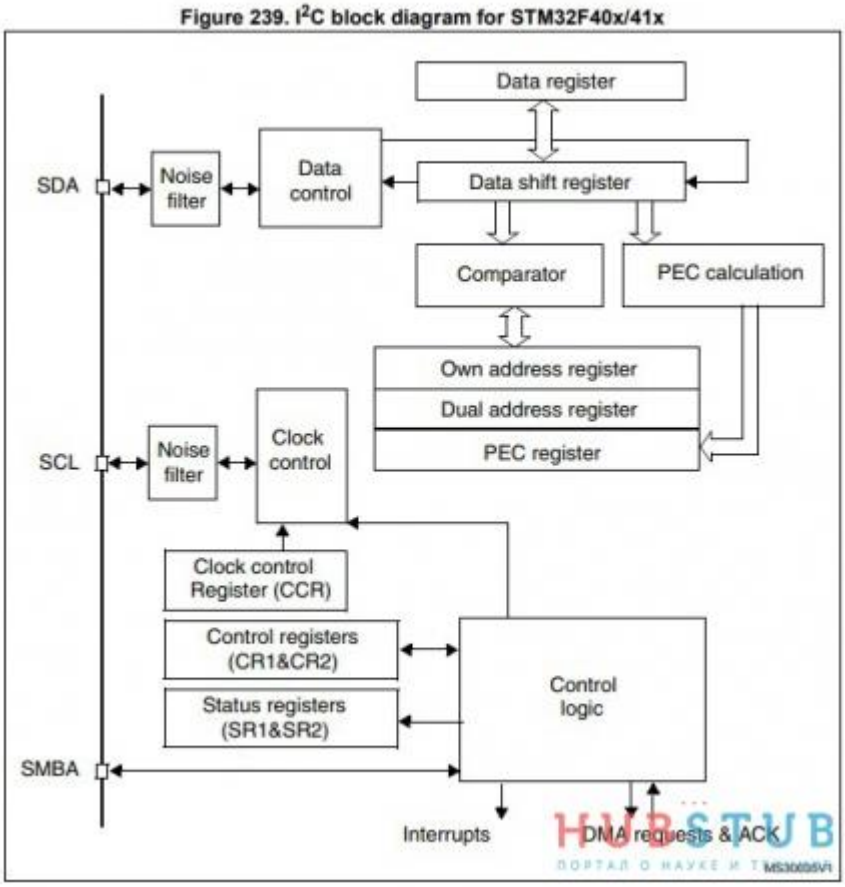



Рисунок 7 – Регистровая модель микроконтроллера

Регистр управления *I2C_CR2* включает в себя:

– *LAST(DMA last transfer)* – единица в этом бите разрешает *DMA* генерировать сигнал окончания передачи *EOT(End of Transfer);*

– *DMAEN(DMA requests enable)* – единица в этом бите разрешает делать запрос к *DMA* при установке флагов *TxE* или *RxNE*;

– *ITBUFEN(Buffer interrupt enable)* – если этот бит сброшен, разрешены все прерывания, кроме прерываний по приему и передаче;

– *ITEVTEN(Event interrupt enable)* – единица в этом бите разрешает прерывания по событию;

– *ITERREN(Error interrupt enable)* – единица в этом бите разрешает прерывания при возникновении ошибок;

– *FREQ[5:0](Peripheral clock frequency)* – в это битовое битовое поле необходимо записать частоту тактирования модуля, она может принимать значение от 2 до 50.

Регистр *I2C_OAR1* состоит из:

– *ADDMODE(Addressing mode)* – этот бит определяет размер адреса *Slave*, ноль соответствует размеру адреса 7 бит, единица – 10 бит;

– *ADD[9:8](Interface address)* – старшие биты адреса, в случае если адрес 10-битный;

– *ADD[1:7](Interface address)* – адрес устройства;

– *ADD0(Interface address*) – младший бит адреса, в случае если адрес 10-битный.

Регистр *I2C_OAR2*:

– *ADD2[7:1]* – альтернативный адрес на который будет отзываться *Slave*;

– *ENDUAL(Dual addressing mode enable)* – единица в этом бите разрешает *Slave* отзываться на альтернативный адрес в 7-битном режиме;

– *I2C_DR* – регистр данных, для отправки данных пишем в регистр *DR*, для приёма читаем его же.

Регистр статуса *I2C_SR1*:

– *SMBALERT(SMBus alert)* – возникает в случае *alert* в шине *SMBus*;

– *TIMEOUT (Timeout or Tlow error)* – возникает если линия *SCL* прижата к земле. Для *master 10mS*, для *slave 25mS*;

– *PECERR (PEC Error in reception)* – возникает при ошибке *PEC (Periodic error connection)* при приеме;

– *OVR (Overrun/Underrun)* – возникает при переполнении данных;

– *AF (Acknowledge failure)* – устанавливается при получении сигнала *NACK*. Для сброса нужно записать 0;

– ARLO *(Arbitration lost (master mode))* – устанавливается при потере арбитража. Для сброса нужно записать 0;

– *BERR (Bus error)* – ошибка шины. Устанавливается в случае возникновения сигнала *START* или *STOP* в неправильный момент;

– *TxE (Data register empty (transmitters))* – устанавливается при опустошении регистра *DR*, а точнее когда данные из него были перемещены в сдвиговый регистр;

– *RxNE (Data register not empty (receivers))* – устанавливается при приеме байта данных, кроме адреса;

– *STOPF(Stop detection (slave mode))* – при работе в режиме *slave* устанавливается при обнаружении сигнала *STOP*, если перед этим был сигнал *ACK*. Для сброса необходимо прочитать *SR1* и произвести запись в *CR1*;

– *ADD10 (10-bit header sent (Master mode))* – устанавливается при отправке первого байта 10-битного адреса;

– *BTF (Byte transfer finished)* – флаг устанавливается по окончании приема/передачи байта, работает только при *NOSTRETCH* равном нулю;

– *ADDR(Address sent (master mode)/matched (slave mode))* – в режиме *master* устанавливается после передачи адреса, в режиме *slave* устанавливается при совпадении адреса. Для сброса нужно прочитать регистр *SR1*, а затем *SR2*;

– *SB(Start bit (Master mode))* – устанавливается при возникновении сигнала *START*. Для сброса флага необходимо прочитать *SR1* и записать данные в регистр *DR* .

Регистр статуса *I2C_SR2*:

– *PEC[7:0](Packet error checking register)* – в это битовое поле записывается контрольная сумма кадра;

– *DUALF(Dual flag (Slave mode))* – ноль в этом бите говорит о том, что адрес который принял *Slave* соответствует *OAR1*, иначе *OAR2*;

– *SMBHOST(SMBus host header (Slave mode))* – устанавливается, когда принят заголовок *SMBus Host*;

– *SMBDEFAULT(SMBus device default address (Slave mode))* – устанавливается, если принят адрес по умолчанию для *SMBus*-устройства;

– *GENCALL(General call address (Slave mode))* – устанавливается, если принят широковещательный адрес в режиме ведомого;

– *TRA(Transmitter/receiver)* – единица в этом бите говорит о том, что модуль работает как передатчик, иначе приемник;

– *BUSY(Bus busy)* – флаг занятости;

– *MSL(Master/slave)* – единица в этом бите говорит о том, что модуль работает в режиме *Master*, иначе *Slave*;

Регистр управления частотой *I2C_CCR*:

– *F/S(I2C master mode selection)* – при установке единицы в этот бит модуль работает в режиме *FAST*, иначе *STANDART*;

– *DUTY (Fm mode duty cycle)* – этот бит задает скважность сигнала *SCL* в режиме *FAST*. Если установлен ноль *tlow/thigh* = 2, иначе *tlow/thigh* = 16/9;

– *CCR[11:0](Clock control register in Fm/Sm mode (Master mode))* – при работе в режиме *Master* задает тактовую частоту линии *SCL*;

– *Sm mode* или *SMBus*;

– *Fm mode*.

Регистр *I2C_TRISE*:

– *TRISE[5:0]* – определяет время нарастания фронта.

– Регистр управления фильтрами *I2C_FLTR*:

– *ANOFF(Analog noise filter OFF)* – ноль в этом бите включает аналоговый фильтр;

– *DNF[3:0](Digital noise filter)* – битовое поле для настройки цифрового фильтра. За подробностями нужно обратиться к документации.

Таким образом была описана регистровая модель I2C микроконтроллера STM32 с описанием каждого регистра.

### **1.2.14** Интерфейс *DCMI*

*DCMI* – это интерфейс цифровой камеры [18]. *DCMI* интерфейс используется для подключения параллельного модуля камеры к *STM32*. Камера генерирует параллельный поток данных вместе с пиксельным тактовым сигналом (*DCMI_PIXCLK*), который позволяет интерфейсу захватывать входящий поток данных. Можно использовать два дополнительных сигнала (*HSYNC* и *VSYNC*) для синхронизации кадра изображения между камерой и микроконтроллером. *DCMI* также поддерживает встроенные коды синхронизации строки/кадра в потоке данных.

*DCMI* позволяет выполнять непрерывный захват. Этот процесс начинается по запросу приложения и продолжается до тех пор, пока бит *CAPTURE* не очистится. В качестве альтернативы снимку, интерфейс позволяет захватить один кадр по запросу приложения. Благодаря функции обрезки интерфейс камеры может вырезать и сохранить прямоугольную часть полученного изображения.

Стандартным способом использования интерфейса камеры является сохранение полученных данных в кадровом буфере в ОЗУ. Далее ядро контроллера может обрабатывать эти данные или передавать их дальше через другой интерфейс (например, *USB* или *Ethernet*).

### **1.2.15** Регистровая модель *DCMI* микроконтроллера с ядром *ARM* Cortex–M4

Аппаратный модуль *DCMI* содержит регистр данных *(DCMI_DR)*, а также десять регистров управления/статуса [19]:

– регистр управления *(DCMI_CR);*

– регистр состояния *(DCMI_SR)*;

– регистр состояния прерываний *(DCMI_RIS)*;

– регистр разрешения прерываний *(DCMI_IER);*

– регистр маски прерываний *(DCMI_MIS)*;

– регистр сброса флагов прерываний *(DCMI_ICR);*

– регистр кодов внутренней синхронизации *(DCMI_ESCR)*;

– регистр сброса маски кодов внутренней синхронизации *(DCMI_ESUR);*

– регистр стартовых значений при захвате части кадра *(DCMI_CWSTRT);*

– регистр величины фрагмента кадра в режиме *CropWindow (DCMI_CWSIZE).*

Блок схема работы *DCMI* представлена на рисунке 8.

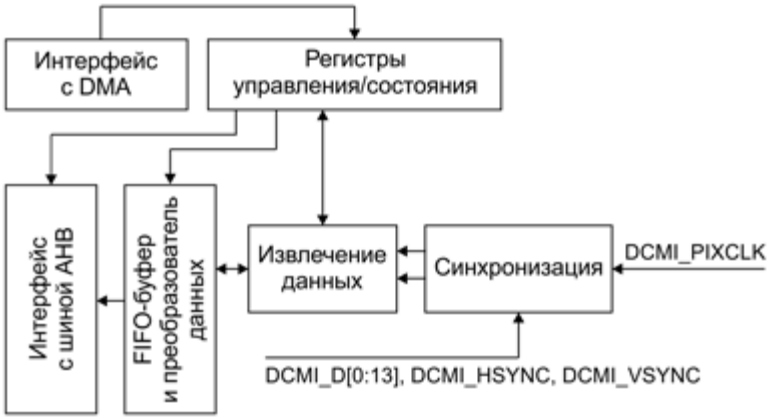



Рисунок 8 – Блок схема *DCMI*

*DCMI* обеспечивает два режима работы: с однократным захватом кадра *(Snapshot mode)* или с потоковым захватом видео *(Continuous mode).* Режим однократного захвата кадра активируется после установки бита *CM* в регистре *DCMI_CR*. Для инициализации захвата необходимо установить бит *CAPTURE* в регистре *DCMI_CR*. Далее *DCIM*-контроллер ждет начала первого кадра и выполняет его захват. После чего бит *CAPTURE* автоматически сбрасывается, а прием следующего кадра не выполняется.

### **1.2.16** Принципы функционирования блока *DMA* прямого доступа к памяти

Прямой доступ к памяти *(DMA)* – это метод, который позволяет устройству ввода-вывода *(I/O)* отправлять или получать данные непосредственно в основную память или из нее, минуя ЦП для ускорения операций с памятью [20].

Определенная часть памяти используется для отправки данных напрямую с периферийного устройства на материнскую плату без участия микропроцессора, чтобы этот процесс не мешал работе компьютера в целом.

В старых компьютерах четыре канала *DMA* были пронумерованы 0, 1, 2 и 3. Когда была введена 16-битная шина расширения промышленного стандарта *(ISA),* были добавлены каналы 5, 6 и 7.

Канал *DMA* позволяет устройству передавать данные, не подвергая ЦП рабочей перегрузке. Без каналов прямого доступа к памяти ЦП копирует каждый фрагмент данных с устройства ввода-вывода, используя периферийную шину. Использование периферийной шины занимает центральный процессор во время процесса чтения/записи и не позволяет выполнять другую работу, пока операция не будет завершена.

С *DMA* ЦП может выполнять другие задачи, пока выполняется передача данных. Передача данных сначала инициируется ЦП. Блок данных может быть передан в память и из памяти с помощью *DMA* тремя способами.

Стандартный прямой доступ к памяти (также называемый сторонним прямым доступом к памяти) использует контроллер прямого доступа к памяти. Контроллер прямого доступа к памяти может создавать адреса памяти и запускать циклы чтения или записи памяти. Он охватывает несколько аппаратных регистров, которые могут быть прочитаны и записаны ЦП.

Эти регистры состоят из регистра адреса памяти, регистра счетчика байтов и одного или нескольких управляющих регистров. В зависимости от функций, предоставляемых контроллером прямого доступа к памяти, эти управляющие регистры могут назначать некоторую комбинацию источника, назначения, направления передачи (чтение с устройства ввода-вывода или запись на него), размер блока передачи и/или количество байт для передачи в одном пакете.

Для выполнения операций ввода, вывода или памяти в память хост-процессор инициализирует *DMA*-контроллер количеством байтов для передачи и используемым адресом памяти. Затем ЦП дает команду периферийному устройству начать передачу данных. Затем контроллер прямого доступа к памяти предлагает адреса и линии управления чтением/записью в системную память. Каждый раз, когда байт данных готовится к передаче между периферийным устройством и памятью, контроллер прямого доступа к памяти увеличивает свой внутренний адресный регистр до тех пор, пока не будет передан полный блок данных.

Прямой доступ к памяти работает по-разному в разных режимах работы:

1 В пакетном режиме полный блок данных передается в непрерывной последовательности. Как только ЦП разрешает контроллеру *DMA* доступ к системной шине, контроллер *DMA* передаст все байты данных в блоке данных, прежде чем передать управление системными шинами обратно ЦП, но это приведет к тому, что ЦП будет неактивен в течение некоторого времени. Этот режим также называется «режимом блочной передачи».

2 Циклический режим используется в системе, в которой невозможно отключить ЦП на время, необходимое для режима пакетной передачи. В

режиме захвата цикла контроллер *DMA* получает доступ к системной шине с помощью сигналов *BR* (запрос шины) и *BG* (предоставление шины), которые аналогичны пакетному режиму. Эти два сигнала управляют интерфейсом между ЦП и контроллером прямого доступа к памяти. С одной стороны, в циклическом режиме скорость передачи блоков данных не такая высокая, как в пакетном режиме, а с другой стороны, время простоя процессора не такое продолжительное, как в пакетном режиме.

3 В прозрачном режиме передача блоков данных занимает больше всего времени, но это также и самый эффективный режим с точки зрения общей производительности системы. В прозрачном режиме контроллер прямого доступа к памяти передает данные только тогда, когда ЦП выполняет операции, не использующие системные шины. Основное преимущество прозрачного режима заключается в том, что ЦП никогда не прекращает выполнение своих программ, а передачи с прямым доступом к памяти бесплатны с точки зрения времени, а недостатком является то, что аппаратному обеспечению необходимо определять, когда ЦП не использует системные шины, что может быть сложным. Это также называется «скрытый режим передачи данных *DMA*».

### **1.2.17** Методика обработки прерывания *DMA*

Прямой доступ к памяти (*DMA*) работает следующим образом [21]:

– устройство, желающее выполнить *DMA*, устанавливает сигнал запроса шины процессора;

– процессор завершает текущий цикл шины, а затем выдает на устройство сигнал предоставления шины;

– затем устройство подтверждает сигнал подтверждения предоставления шины;

– процессор улавливает изменение состояния сигнала подтверждения предоставления шины и начинает прослушивать данные и адресную шину на предмет активности контроллера-*DMA;*

– контроллер-*DMA* выполняет передачу с адреса источника на адрес назначения.

Во время этих передач процессор отслеживает адреса на шине и проверяет, кэшируются ли в процессоре какие-либо адреса, измененные во время операций прямого доступа к памяти. Если процессор обнаруживает кэшированный адрес на шине, он может предпринять одно из двух действий: сделать недействительной запись внутреннего кэша для адреса, участвующего в операции записи или обновить внутренний кэш при обнаружении записи.

После завершения операций прямого доступа к памяти устройство освобождает шину, подавая сигнал освобождения шины. Процессор

подтверждает освобождение шины и возобновляет свои циклы шины с того места, где он был остановлен.

Непосредственно обработка прерываний может выполняться по двум сценариям – когда аппаратное обеспечение не поддерживает идентификацию устройства, инициировавшего прерывание, и когда поддерживает.

В случаях, когда идентификация устройства не поддерживается на аппаратном уровне, возможные прерывающие устройства должны быть опрошены программно:

– устройство устанавливает сигнал прерывания на аппаратно-запрограммированном уровне прерывания;

– процессор регистрирует прерывание и ожидает завершения выполнения текущей инструкции;

– как только выполнение текущей инструкции завершено, процессор инициирует обработку прерывания, сохраняя содержимое текущего регистра в стеке;

– затем процессор переключается в режим супервизора и инициирует цикл подтверждения прерывания;

– ни одно устройство не отвечает на цикл подтверждения прерывания, поэтому процессор выбирает вектор, соответствующий уровню прерывания;

– адрес, найденный в векторе, является адресом процедуры обслуживания прерываний (*ISR*);

– *ISR* опрашивает все устройства, чтобы найти устройство, вызвавшее прерывание. Это достигается путем проверки регистров состояния прерывания на устройствах, которые могли инициировать прерывание;

– как только устройство обнаружено, управление передается обработчику, специфичному для прерывающего устройства;

– после того, как специфичная для устройства подпрограмма *ISR* выполнила свою работу, *ISR* выполняет инструкцию «возврата из прерывания».

Выполнение команды «возврат из прерывания» приводит к восстановлению состояния процессора. Процессор возвращается в пользовательский режим.

В случаях, когда идентификация устройства поддерживается на аппаратном уровне, возможные прерывающие устройства идентифицируются на аппаратном уровне:

– устройство устанавливает сигнал прерывания на аппаратно-запрограммированном уровне прерывания;

– процессор регистрирует прерывание и ожидает завершения выполнения текущей инструкции;

– как только выполнение текущей инструкции завершено, процессор инициирует обработку прерывания, сохраняя содержимое текущего регистра в стеке;

– затем процессор переключается в режим супервизора и инициирует цикл подтверждения прерывания;

– прерывающее устройство отвечает на цикл подтверждения прерывания номером вектора для прерывания;

– процессор использует номер вектора, полученный выше, и выбирает вектор;

– адрес, найденный в векторе, является адресом процедуры обслуживания прерываний (*ISR*) прерывающего устройства.

После того, как подпрограмма *ISR* выполнила свою работу, *ISR* выполняет команду «возврата из прерывания».

Выполнение команды «возврат из прерывания» приводит к восстановлению состояния процессора. Процессор возвращается в пользовательский режим [22].

### **1.2.18** Структура и логика функционирования цифровой видеокамеры на базе процессора *OV9655*

*OV9655 CameraChip* представляет собой датчик изображения, обеспечивающий полную функциональность одночиповой камеры *SXGA* (1280x1024) и процессора изображений в одном корпусе. *OV9655* обеспечивает полнокадровый, субдискретизированный, масштабированный или оконный режим изображения в широком диапазоне форматов, управляемый через интерфейс последовательной шины управления камерой (*SCCB*) [22]. Камера способна работать в режиме 15 кадров в секунду (*fps*) в разрешении *SXGA* с полным пользовательским контроль над качеством изображения, форматированием и передачей выходных данных.

Вся необходимая обработка изображений функции, такая как управление экспозицией, гамма, баланс белого, насыщенность цвета, регулировка оттенка, шумоподавление и многое другое, также программируется через интерфейс *SCCB*. Структура *OV9655* приведена на рисунке 9.

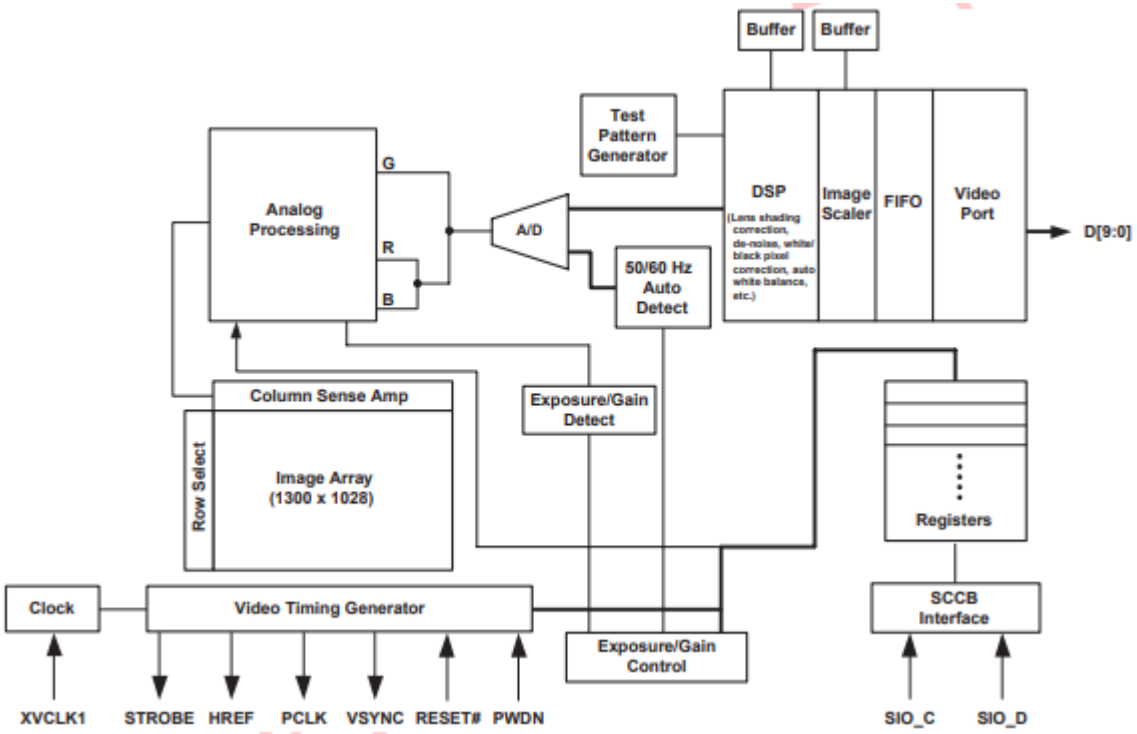



Рисунок 9 – Структура *OV9655*

Как можно увидеть на схеме, *OV9655* содержит в себе следующие компоненты:

– массив пикселей;
– аналоговый обработчик сигналов;
– аналого-цифровые преобразователи;
– цифровой обработчик сигналов;
– блок форматирования выхода;
– синхронизирующий генератор;
– интерфейс *SCCB*;
– цифровой видео интерфейс.

Массив пикселей определяет максимальное разрешение камеры, а именно 1300 столбцов на 1028 строк (всего 1336400 пикселей).

Синхронизирующий генератор управляет генерацией кадров, генерацией внутренних сигналов синхронизации, синхронизацией частоты кадров, контролем экспозиции, а также выходами внешней синхронизации.

Аналоговый обработчик сигналов – блок, который выполняет управление экспозицией и балансом белого у аналогового изображения.

Цифровой обработчик сигналов – блок, который управляет изменением данных в *RGB* и некоторым контролем качества изображения: устранять цветовые помехи, контролировать оттенок и насыщенность, шумоподавление, удаление белых пикселей.

Блок форматирования выхода управляет форматированием данных перед отправкой изображения (например, уменьшение размера изображения).

Интерфейс последовательной шины управления камерой (*SCCB*) контролирует все происходящие операции.

### **1.2.19** Физический и канальный уровни интерфейса *MII*

*MII* (независимый от среды интерфейс) – это стандарт, используемый для соединения блока *MAC* (управление доступом к среде) с физическим уровнем *PHY* для сетевых устройств [23]. Эти две части общего сетевого устройства *Ethernet* выполняют разные функции в рамках модели *OSI* (межсетевого взаимодействия открытых систем). Каждый уровень системы выполнял следующие функции:

1 *PHY* (физический уровень) – это микросхема, которая преобразует цифровые данные из *MAC* и отправляет их по физическому сетевому интерфейсу в виде аналогового сигнала. Эти микросхемы функционируют как приемопередатчики, поэтому они модулируют аналоговые сигналы, передаваемые по физическому уровню. При подключении к физическому каналу по *Ethernet* выходные данные *PHY* отправляются на приемопередатчик для преобразования модулированных аналоговых сигналов в цифровой сигнал.

2 *MAC* работает на логическом уровне и функционирует как интерфейс между *CPU/FPGA/MCU/ASIC* для обработки данных и связи с чипом *PHY*. *MAC* обеспечивает необходимые возможности обработки данных, а также отправляет данные и получает данные от *PHY*.

Стандарт *MII* передает 4-битные блоки данных между *MAC* и *PHY* для передачи данных *TX* и *RX*. *PHY* работает на частоте 2,5 МГц (режим 10 Мбит/с) или 25 МГц (режим 100 Мбит/с). Связь в *MII* не является двунаправленной, поэтому определенные сигналы разделяются на наборы сигналов *TX* и *RX*. Тактовый сигнал, используемый для управления *PHY*, также используется для того, чтобы *MAC* отправлял *TX*-данные на *PHY*; данные отправляются с *MAC* на *PHY* по переднему фронту этого импульса, что позволяет передавать данные синхронно.

Микросхемы *PHY* также используются в других протоколах связи, таких как *USB, SATA* и беспроводная локальная сеть/*Wi-Fi*. Некоторые функции могут быть интегрированы в уровень *MAC*, в зависимости от соответствующих приложений. В *Ethernet* количество сигналов, необходимых для связи *PHY* с *MAC*, довольно велико в соответствии со стандартом *MII*, поэтому был разработан стандарт *RMII* для уменьшения количества сигналов.

### **1.2.20** Физический и канальный уровни интерфейса *RMII*

Что касается *RMII*, то разница с *MII* лишь в тактовой частоте и количестве сигналов для связи с *MAC*.

Каждая микросхема *PHY* управляет одним физическим интерфейсом, поэтому печатные платы для таких устройств, как сетевые коммутаторы, содержат множество каналов для обеспечения связи между *PHY* и *MAC*. В *MII* каждому *PHY* требуется 18 сигналов для связи с *MAC*, и только 2 из этих сигналов могут совместно использоваться несколькими устройствами *PHY*. Поэтому *RMII* (сокращенный MII) был разработан как вариант *MII*, чтобы вдвое сократить количество неразделяемых сигналов на *PHY*-интерфейс до 8.

Интерфейс *RMII* также способен поддерживать скорость передачи данных 10 Мбит/с и 100 Мбит/с, а также существуют варианты с поддержкой 1 гигабит. В *RMII* тактовая частота, используемая в *PHY*, постоянно работает на уровне 50 МГц для скоростей передачи данных как 10 Мбит/с, так и 100 Мбит/с. Это удвоение тактовой частоты при 100 Мбит/с также позволяет вдвое сократить количество сигналов для связи между *PHY* и *MAC*. Всего для связи требуется 9 сигналов, из которых до 3 могут быть разделены между несколькими *PHY*.

### **1.2.21** Структура и логика функционирования микросхемы *LAN8720*

*LAN8720* – это приемопередатчик физического уровня (*PHY*) *10BASE-T/100BASE-TX* с низким энергопотреблением и переменным напряжением ввода-вывода, соответствующий стандартам *IEEE 802.3-2005*.

*LAN8720* поддерживает связь с *Ethernet MAC* через стандартный интерфейс *RMII*. Он содержит полнодуплексный приемопередатчик *10-BASE-T/100BASE-TX* и поддерживает работу на скоростях 10 Мбит/с (*10BASE-T*) и 100 Мбит/с (*100BASE-TX*). В *LAN8720* реализовано автоматическое согласование для определения наилучшей возможной скорости и дуплексного режима работы. Поддержка *HP Auto-MDIX* позволяет использовать прямые или перекрестные кабели *LAN*. *LAN8720* поддерживает как функции регистрации, соответствующие стандарту *IEEE* 802.3-2005, так и функции регистрации, зависящие от поставщика. Однако для работы не требуется доступ к регистру. Выбираемые регистром параметры конфигурации могут использоваться для дальнейшего определения функциональности трансивера. В соответствии со стандартами *IEEE* 802.3-2005 все контакты цифрового интерфейса рассчитаны на напряжение 3,6 В. Устройство может быть настроено для работы от одного источника питания 3,3 В с использованием встроенного линейного регулятора от 3,3 В до 1,2 В. Линейный регулятор может быть опционально отключен, что позволяет использовать высокоэффективный внешний регулятор для снижения рассеиваемой мощности системы [24].

Структура *LAN8720* приведена на рисунке 10.

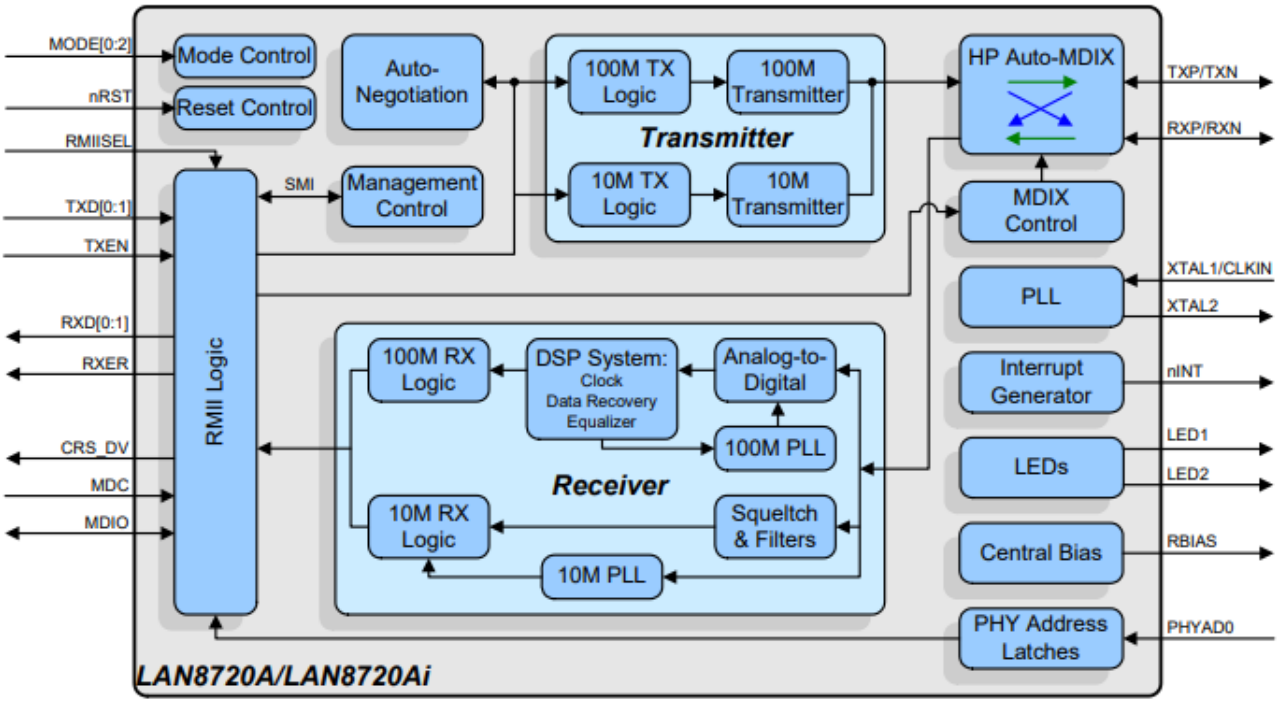



Рисунок 10 – Структурная схема *LAN8720*

Функции, которые может выполнять *LAN8720*, можно классифицировать следующим образом:

– трансивер;

– автосогласование;

– Поддержка *HP Auto-MDIX*;

– *MAC*-интерфейс;

– последовательный интерфейс управления (*SMI*);

– управление прерываниями;

– прочие функции.

Данное устройство можно использовать в качестве *MAC*-интерфейса.

Устройство поддерживает независимый от среды интерфейс (*RMII*) с малым количеством контактов. Интерфейс *RMII* имеет следующие характеристики:

– способен поддерживать скорость передачи данных 10 Мбит/с и 100 Мбит/с;

– один опорный тактовый сигнал используется как для передачи, так и для приема;

– обеспечивает независимые 2-битные пути передачи и приема данных;

– использует уровни сигналов *LVCMOS*, совместимые с обычными цифровыми процессами *CMOS ASIC*.

*RMII* включает следующие интерфейсные сигналы (1 опционально):

– передавать данные – *TXD*[1:0];
– передать строб – *TXEN*;
– получить данные – *RXD*[1:0];
– получить ошибку – *RXER* (необязательно);
– определение несущей – *CRS_DV*;
– опорный тактовый сигнал *REF_CLK*.

*CRS_DV* утверждается устройством, когда принимающая среда не простаивает. *CRS_DV* устанавливается асинхронно во время обнаружения несущей по критериям, относящимся к режиму работы. В режиме *10BASE-T*, когда шумоподавление пройдено, или в режиме *100BASE-X*, когда обнаруживаются 2 несмежных нуля в 10 битах, считается, что несущая обнаружена.

Потеря несущей должна привести к отмене подтверждения *CRS_DV* синхронно с циклом *REF_CLK*, который представляет первую пару битов полубайта на *RXD*[1:0].

### **1.2.22** Организация *LwIP*–стека

*LwIP* – это небольшая независимая реализация набора протоколов *TCP/IP*, первоначально разработанная Адамом Дункельсом. Целью реализации *lwIP* является сокращение использования ресурсов при сохранении полномасштабного *TCP* [25]. Это делает *lwIP* пригодным для использования во встроенных системах с десятками килобайт свободной оперативной памяти и местом для примерно 40 килобайт кода.

*LwIP* поставляется со следующими протоколами:

– *IPv4* и *IPv6*, включая пересылку пакетов через несколько сетевых интерфейсов;
– *ICMP* для обслуживания и отладки сети;
– *IGMP* для управления многоадресным трафиком;
– *MLD* (обнаружение прослушивателя многоадресной рассылки для *IPv6*);
– *ND* (обнаружение соседей и автоматическая настройка адресов без сохранения состояния для *IPv6*);
– *DHCP* и *DHCPv6*;
– *UDP* (протокол пользовательских дейтаграмм);
– *TCP* (протокол управления передачей);
– *API* для повышения производительности;
– *TLS*: дополнительный многоуровневый *TCP* для почти прозрачного *TLS* для любого протокола на основе *TCP*;
– *PPPoS* и *PPPoE*;
– *DNS* (преобразователь доменных имен, включая *mDNS*);

– *6LoWPAN* – стандарт взаимодействия по протоколу *IPv6* поверх маломощных беспроводных персональных сетей.

Организация *LwIP* стека представлена ниже на рисунке 11.

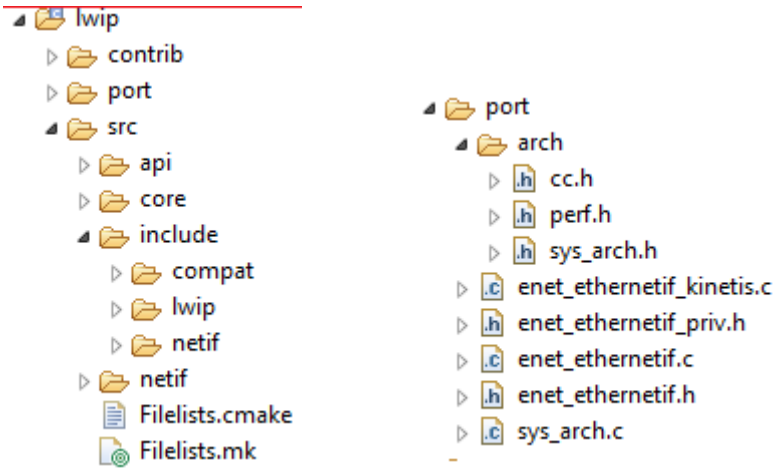



Рисунок 11 – Организация *LwIP*

*Enet_ethernetif.c/h* нужны для адаптации стека *LwIP* к базовому драйверу *Ethernet MCUXpresso SDK*, предоставляя интерфейсы *Ethernet phy*, *init* и *ethernetif_input.*

*Cc.c/h* представляет собой подсказки компилятора для упаковки и специфичные для платформы диагностические данные.

*Perf.h* нужен для измерения производительности для конкретной архитектуры.

Папка *src* содержит последний стабильный исходный код *LwIP*:

– *api* – файлы *netconn* и *API* сокетов;

– *core* – основные файлы *LwIP;*

– *include* – включаемые файлы *LwIP;*

– netif – файлы сетевого интерфейса.

*LwIP* предоставляет программистам три *API*, которые они могут использовать для связи с кодом *TCP/IP*:

1 *Raw API*. Это управляемый событиями *API*, предназначенный для использования без операционной системы, реализующей отправку и получение без копирования. *Raw API* – это родной интерфейс *LwIP*. Он предусматривает использование обратных вызовов функций (*callbacks*) внутри стека. Это означает, что вам перед началом работы со стеком необходимо присвоить указатели на функции-обработчики событий, которые в процессе работы будут вызываться внутри *LwIP*. *Raw API* имеет наибольшую производительность и наименьший результирующий размер кода.

2 *Netconn API* – высокоуровневый последовательный интерфейс, построенный поверх *Raw API*. Этот интерфейс требует наличия *RTOS* и поддерживает мнопоточные операции.

3 *BSD Socket API* – высокоуровневый интерфейс сокетов, разработанный поверх *Netconn API*. Этот интерфейс обеспечивает высокую переносимость ваших приложений, поскольку является стандартизированным *API*.

*LwIP* получил очень широкое распространение во встраиваемых системах на базе микроконтроллеров благодаря низкому потреблению оперативной памяти. Именно этот *TCP/IP* стек используется в фреймворке *ARM mbed* и генераторе кода инициализации *STM32CubeMX*.

### **1.2.23** Стандарты сжатия данных в *IP*-видеонаблюдении

Ранее были введены такие понятия, как пиксель и массив пикселей (кадр). К слову, на данный момент в системах видеонаблюдения самые распространённые размеры кадров: 960x576 (*WD1*), 1280x720 (*HD*), 19 20x1080 (*FullHD*), 2688x1520 (*4Mpix*) и 2560x1920 (5 *Mpix*).

Когда кадры чередуются с некоторой частотой, получается видео. Частота, при которой человеческий глаз воспринимает такой видеопоток плавным – 24 кадра в секунду.

Также видеопоток характеризуется битрейтом – количеством бит информации, которое используется для хранения или для передачи аудио или видео. Битрейт может быть как постоянным, так и переменным. Постоянный битрейт соответствует заданным параметрам и остаётся неизменным на протяжении всего файла. Его главное достоинство в том, что можно предсказать размер конечного файла. При переменном битрейте кодек выбирает его значение, исходя из параметров желаемого качества. В течение всего кодируемого видеофрагмента битрейт может изменяться.

Итак, зачем сжимать видео? Если, например, взять видеопоток с разреением 1920x1080 пикселей со скоростью 2 кадра в секунду, а также каждый пиксель кодировать в *RGB24* (то есть 24 бита на пиксель), то на один кадр уйдёт 47.5 Мбит/с пропускной способности канала, а на 24 таких кадра – 1140 Мбит/с. В свою очередь такая же последовательность кадров, но сжатая в соответствии со стандартом *H*.264 будет занимать в среднем в 150 раз меньше пропускной способности канала, а также места в хранилище.

В контексте видеонаблюдения имеет смысл рассматривать поколение кодеков семейства *H*, так как они нацелены на уменьшение потока цифрового видео по сети [26].

На данный момент в системах видеонаблюдения долгое время доминирует алгоритм сжатия *H*.264, который заключается в исключении избыточных данных и сокращении их объема по различным алгоритмам.

При настройке кодирования в системах видеонаблюдения встречаются три основных профиля кодека *H*.264:

– *Baseline* профиль, который подразумевает минимальную нагрузку на процессор декодирующего устройства при несильном сжатии. Предназначен для просмотра видео с видеокамеры на внешнем сервере.

– *Main* профиль, который создаёт среднюю нагрузку на процессор при сильном сжатии.

– *High* профиль обеспечивает максимальное сжатие с сильной нагрузкой на устройство декодирования. Битрейт при работе с таким профилем будет в 2-3 раза ниже, чем при использовании *baseline* профиля.

Формат сжатия *H.265 High Efficiency Video Coding (HEVC)* стал значительным шагом вперед в области кодирования цифрового видеосигнала, главным преимуществом которого является почти в 2 раза увеличенная эффективность по сравнению с предшествующим стандартом *H*.264. То есть благодаря новому алгоритму для передачи сигнала требуется вдвое меньшая пропускная способность сети, а для хранения вдвое меньшая ёмкость накопителей.

Параллельное кодирование, предусмотренное стандартом *H*.265, даёт возможность одновременной обработки разных частей кадра, что существенно ускоряет воспроизведение и даёт возможность в полной мере использовать свойство новых процессоров – многоядерность.

Кроме этого, новый стандарт получил технологию произвольного доступа к изображению (*Clean Random Access*), которая позволяет произвести декодирование случайно выбранного кадра без необходимости обработки предыдущих в потоке изображений. Это особенно желательно, когда при мониторинге требуется оперативно переключиться на определённый канал.

Несмотря на все преимущества, *H*.265 ещё далёк от повсеместного использования. Во-первых, из-за того, что для его использования необходима обновлённая аппаратная часть, во-вторых, чтобы использовать кодек необходима покупка патента.

Кроме двух вышеприведённых кодеков, существует *H*.264+, который был разработан специально под нужны видеонаблюдения.

На видео, полученном с охранных видеокамер, сцена всегда постоянна и практически не изменяется, представляющие интерес подвижные объекты могут отсутствовать на протяжении длительного времени, а шумы, возникающие в плохих условиях освещения, ощутимо влияют на качество изображения. В обновлённом формате все эти особенности были учтены и обрабатываются следующими технологиями, повышающими степень сжатия:

– кодирование с предсказанием на основе модели фона;

– шумоподавление;

– долгосрочное управление видеопотоком.

Кодирование с предсказанием используется тогда, когда фон в видеонаблюдении стабилен, то этот фон лучше всего использовать в качестве опорного кадра, тем самым повысить эффективность сжатия неподвижных объектов и снизить поток данных, приходящийся на опорные кадры. Интеллектуальный алгоритм предсказания выбирает опорные кадры среди тех, в которых меньше всего движущихся объектов.

Шумоподавление в формате *H*.264+ с помощью специальных алгоритмов фон отделяется от движущегося объекта и кодируется с более высокой степенью сжатия. Такая технология позволяет частично подавлять шумы и уменьшать битрейт.

Долгосрочное управление видеопотоком. Формат *H*.264+ имеет алгоритмы отслеживания интенсивности видеопотоков и в зависимости от времени суток автоматически изменяет степень сжатия. Такая технология управления видеопотоком позволяет не только уменьшить объём видеоархива, но и сохранить качество изображения движущихся объектов.

### **1.2.24** Технология PoE

*Power over Ethernet (PoE)* – это стандарт, который способен одновременно передавать данные и питание по одному сетевому кабелю.

Технология *PoE* отправляет данные со скоростью 10/100/1000 Мбит/с и может выдавать мощности 15 Вт, 30 Вт, 60 Вт и до 90 Вт на устройства категорий *Cat5e*, *Cat6*, *Cat6a*. Кабели *Cat7* и *Cat8 Ethernet* на максимальное расстояние 100 м.

Технология *PoE* основана на стандартах *IEEE* 802.*3af,* 802.*3at* и 802.*3bt,* установленных Институтом инженеров по электротехнике и электронике, и определяет, как должно работать сетевое оборудование для обеспечения взаимодействия между устройствами [27].

Устройствами с поддержкой *PoE* могут быть как источником питания (*PSE*), так и с принимающими элементами питания (*PD*), а иногда и то, и другое. Устройство, передающее питание, называется *PSE*, а устройство, на которое подается питание, – это *PD*.

Как правило, данная технология используется в *IP*-телефонии, в беспроводных точках доступа и в *IP*-видеонаблюдении.

В технологии *PoE* часто используют разные пары выводов питания. Эти два метода известны как «Режим *A*» и «Режим *B*».

В режиме *A* пара контактов данных 1-2 образует одну сторону источника постоянного тока, а пара контактов 3-6 – другую сторону, оставляя пары выводов 4-5 и 7-8 неиспользованными. Устройства, использующие режим *A*, иногда называют «конечными».

В отличие от режима *A*, режим *B* не оставляет неиспользованными пары выводов данных. Пары контактов 1-2 и 3-6 отправляют данные. Пара

контактов 4-5 образует одну сторону источника постоянного тока, а пара контактов 7-8 – другую. Устройства, использующие режим *B*, также называются «промежуточными» устройствами.

Хотя различия между режимом *A* и режимом *B* минимальны, их важно учитывать, поскольку игнорирование того, какие контакты питания используются для передачи и приема питания, может привести к неработающим соединениям.

Ограничений в использовании данной технологии немного:

– работает на дистанциях до 100 м;

– несовместимые устройства требуют дополнительного оборудования;

– небольшая мощность.

По технологии *PoE* можно передавать сигнал до 100 метров от источника к приёмнику независимо от того, куда подается питание. Стандарты кабелей *Ethernet* ограничивают общую длину кабелей до 100 м, поэтому данное ограничение в большинстве случаев ничего не значит.

В случае с *IP*-видеокамерами с поддержкой *PoE* питание подаётся посредством порта *RJ*-45, а источником питания служит *PoE*-коммутатор или *PoE*-инжектор, используемый в некоторых сетях в качестве промежуточного питающего оборудования. Согласно задействованному стандарту через вышеописанный современный *Ethernet*-кабель подаётся напряжение в 45В, что обеспечивает на выходе мощность в 15 Ватт [28]. Благодаря тому, что напряжение достаточно высоко, отпадает необходимость использовать кабели с большим сечением проводника, что удешевляет технологию.

### **1.2.25** Принципиальные основы и схемы зарядки литий–ионных аккумуляторных батарей

Литий-ионная аккумуляторная батарея в общем случае состоит из анода, катода, сепаратора, электролита и двух токосъемников (положительного и отрицательного). Анод и катод хранят литий. Электролит переносит положительно заряженные ионы лития от анода к катоду и обратно через сепаратор. Движение ионов лития создает свободные электроны в аноде, которые создают заряд на коллекторе положительного тока. Затем электрический ток течет от токоприемника через питаемое устройство (мобильный телефон, компьютер и т. д.) к отрицательному токоприемнику. Сепаратор блокирует поток электронов внутри аккумулятора [29].

В то время как батарея разряжается и обеспечивает электрический ток, анод выпускает ионы лития к катоду, создавая поток электронов с одной стороны на другую. При включении устройства происходит обратное: ионы лития высвобождаются катодом и принимаются анодом.

Двумя наиболее распространенными понятиями, связанными с батареями, являются плотность энергии и плотность мощности. Плотность

энергии измеряется в ватт-часах на килограмм (Втч/кг) и представляет собой количество энергии, которое батарея может хранить по отношению к ее массе. Плотность мощности измеряется в ваттах на килограмм (Вт/кг) и представляет собой количество энергии, которое может быть выработано батареей по отношению к ее массе. Чтобы нарисовать более ясную картину, подумайте об осушении бассейна. Плотность энергии аналогична размеру бассейна, а плотность мощности сравнима с максимально быстрым сливом бассейна.

Литий-ионные аккумуляторы производят как в корпусном, так и в ламинированном исполнении (гель-полимерные), электроды и электродные массы которых помещены в герметичный пакет из специальной пленки. Электрохимические процессы протекают одинаково как в тех, так и в других.

Классический способ заряда *Li-ion* аккумулятора делится на два этапа [30]. Первый – это заряд постоянным током, второй – заряд при постоянном напряжении (рисунок 12).

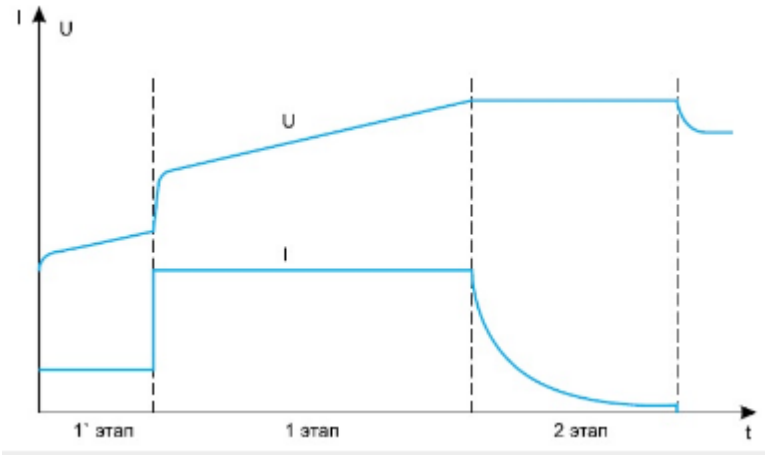



Рисунок 12 – Этапы заряда *Li-ion* аккумулятора

Этап, который идёт перед первым этапом, необходим, когда напряжение на аккумуляторе ниже некоторого установленного значения (например, 2,5 В). При долгом хранении аккумулятора вследствие саморазряда и/или потребления системы обеспечения функционирования (СОФ) напряжение на аккумуляторе может упасть ниже, к примеру, 2,5 В (СОФ входит в состав аккумуляторной батареи, даже если она состоит из одного аккумулятора). Малый ток заряда обеспечивает постепенный выход активных электродных материалов на заданные уровни напряжения, при которых они штатно функционируют (например, при более 2,8 В), после чего включается основной ток заряда. Данный режим призван обеспечить более долгую жизнь аккумулятора при выходе его из заданного диапазона напряжений. Также этап 1′ применяется при заряде аккумулятора при низких температурах, например ниже +5 °С – для «разогрева» электродных масс.

На первом этапе заряд осуществляется номинальным током, который измеряется в долях от номинальной емкости аккумулятора (Сн). Например, емкость аккумулятора 10 А·ч, номинальный ток заряда 0,2Сн, то есть 2 А – пятичасовой режим заряда. Понятно, что потребитель хочет, чтобы заряд осуществлялся как можно быстрее – в течение 1-2 часов, что соответствует 0,5-1Сн. Такой режим заряда обычно называют ускоренным. Для нормальной работы аккумулятора номинальный ток заряда лежит в пределах 0,2-0,5Сн, а ускоренный, – в диапазоне 0,5-1Сн.

Второй этап – заряд при постоянном напряжении и падающем токе. Ток на этом этапе падает до определенного значения. Например, процесс считается завершенным при установлении тока заряда менее 0,1-0,05Сн. Как было показано выше, продолжительность фазы падающего тока зависит от тока заряда. Для номинального режима заряда (0,2Сн) она длится обычно не более нескольких десятков минут, при этом аккумулятор набирает до 0,1–0,15Сн. Время заряда падающим током также зависит от степени деградации аккумулятора в процессе эксплуатации (иначе говоря, от срока службы и количества циклов заряд/разряд). Чем больше деградация, тем длиннее фаза падающего тока.

Производители электроники предоставляют уже готовые схемотехнические решения, реализующие описанный выше алгоритм заряда, выполненные в одном корпусе микросхемы – например *LTC4058* и *BQ24295*, которые будут описаны в следующей главе.

### **1.2.26** Структура и логика функционирования микросхем *LTC4058* и *BQ24295* зарядки литий-ионных аккумуляторных батарей

*LTC4058* – это комплексное линейное зарядное устройство постоянного тока/постоянного напряжения для одноэлементных литий-ионных аккумуляторов [31]. Пакет *DFN* и небольшое количество внешних компонентов делают *LTC4058* хорошим вариантом для портативных устройств. Кроме того, *LTC4058* предназначен для работы через *USB*-интерфейс. *LTC4058* может определять температуру аккумулятора по шкале Кельвина для более точной зарядки с плавающим напряжением. Термическая обратная связь регулирует поступающий ток для ограничения температуры аккумулятора во время работы при работе с высокой мощностью или в условиях высокой температуры. Напряжение заряда фиксируется на уровне 4,2 В, а поступающий ток контролируется резистором. Данная микросхема завершает цикл заряда, когда ток заряда падает до 10% от запрограммированного значения после того, как будет получен нужный уровень заряда. Когда источник питания (сетевой адаптер или источник питания *USB*) убран, *LTC4058* входит в состояние низкого тока, сбрасывая ток разряда батареи менее 2 мкА.

Схема *LTC4058* представлена на рисунке 13.

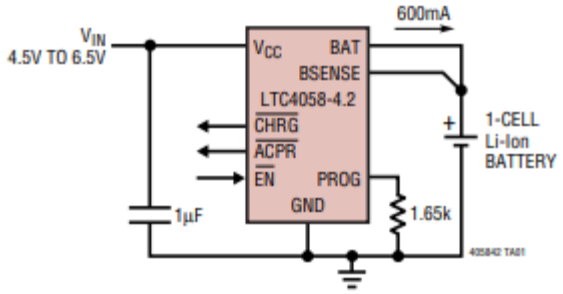



Рисунок 13 – Схема *LTC4058*

*Bq24295* – это устройство управления зарядом аккумулятора с переключением и управления питанием системы для одноэлементных литий-ионных и литий-полимерных аккумуляторов в широком спектре устройств. Питание с низким сопротивлением оптимизирует эффективность работы в режиме переключения, сокращает время зарядки батареи и продлевает срок службы батареи во время фазы разрядки [32].

Последовательный интерфейс *I2C* с зарядкой и системными настройками делает устройство действительно гибким решением. Устройство поддерживает источники входного сигнала *USB* 3,9–6,2 В, включая стандартный *USB*-хост-порт и *USB*-порт для зарядки с защитой от перенапряжения 6,4 В.

*Bq24295* соответствует спецификациям питания *USB* 2.0 и *USB* 3.0 с регулированием входного тока и напряжения. Чтобы установить предел входного тока по умолчанию, *bq24295* определяет источник входного сигнала с помощью обнаружения *D+/D-* в соответствии со спецификацией зарядки аккумулятора по *USB*. Кроме того, *bq24295* обнаруживает нестандартные адаптеры 2A/1A. *Bq24295* поддерживает режим форсирования батареи, подавая регулируемое напряжение 4,55–5,5 В (по умолчанию 5,1 В) на вывод *PMID* с минимальным током 1,5 А.

Схема *Bq24295* представлена на рисунке 14.

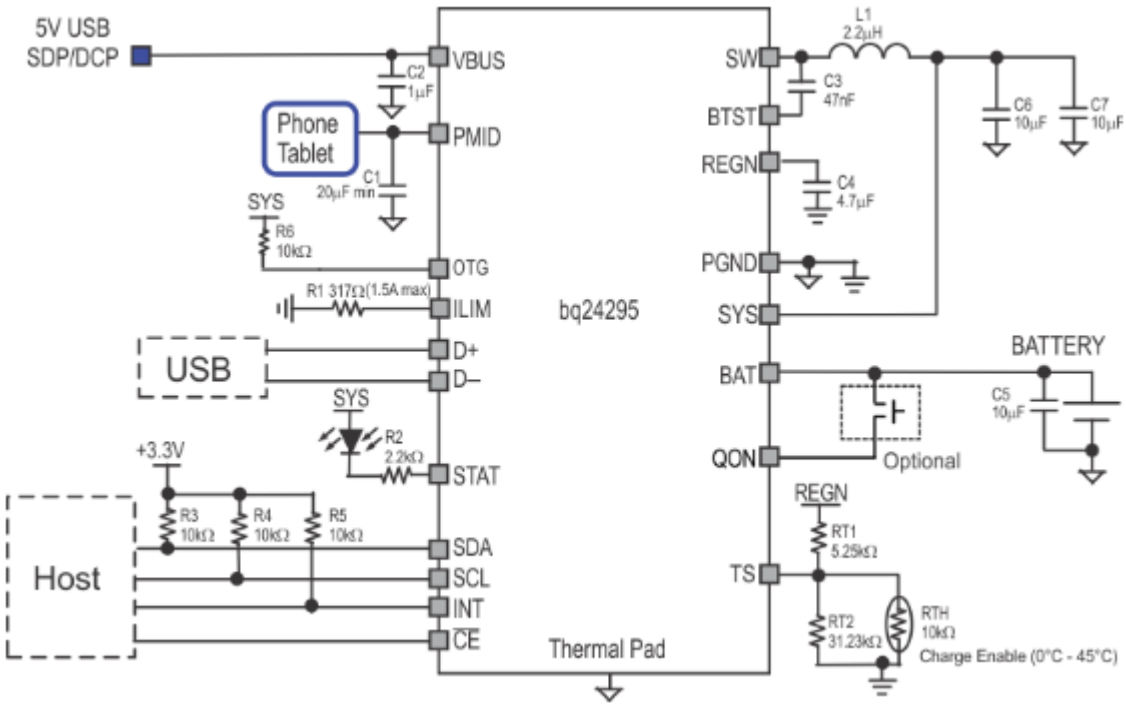



Рисунок 14 – Схема *BQ24295*

Управление цепью питания регулирует напряжение системы немного выше напряжения батареи, но не падает ниже минимального напряжения системы 3,5 В (программируется). Благодаря этой функции система продолжает работать, даже если батарея полностью разряжена или удалена. Когда достигается предел входного тока или предел напряжения, управление цепью питания автоматически снижает ток заряда до нуля. По мере того, как нагрузка на систему продолжает расти, цепь питания разряжает батарею до тех пор, пока не будут удовлетворены требования к мощности системы. Эта работа в дополнительном режиме предотвращает перегрузку источника входного сигнала.

Устройства инициируют и завершают цикл зарядки без программного управления. Микросхема автоматически определяет напряжение батареи и заряжает батарею в три этапа: предварительный этап, этап с постоянным током и этап с постоянным напряжением. В конце цикла зарядки микросхема автоматически останавливается, когда ток заряда ниже заданного предела в фазе постоянного напряжения. Когда полный заряд батареи падает ниже порога перезарядки, микросхема автоматически начинает новый цикл зарядки.

Устройства обеспечивают различные функции безопасности для зарядки аккумулятора и работы системы, включая контроль температуры. таймер безопасности зарядки и защиту от перенапряжения/перегрузки по току. Терморегуляция снижает зарядный ток, когда температура перехода превышает 120°C (программируется).

## 2 РАЗРАБОТКА СТРУКТУРНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ *IP*–ВИДЕОКАМЕРЫ

В Единой системе конструкторской документации выделяют несколько видов электрических схем:

1 Схема структурная, которая определяет функциональные части устройства.

2 Схема функциональная, которая определяет основные процессы, которые происходят во время работы устройства.

3 Схема принципиальная, которая определяет полный состав элементов и взаимосвязи между ними и, как правило, дающий полное (детальное) представления о принципах работы устройства.

4 Схема монтажная, которая показывает соединения составных частей устройства, и которая определяет объекты, которыми осуществляются эти соединения.

5 Схема общая, которая определяет составные части комплекса устройства.

6 Схема расположения, которая определяет относительное расположение составных частей устройства.

7 Схема объединенная – содержит элементы различных типов схем одного вида.

На рисунке 15 приведена структурная схема *IP*-камеры.

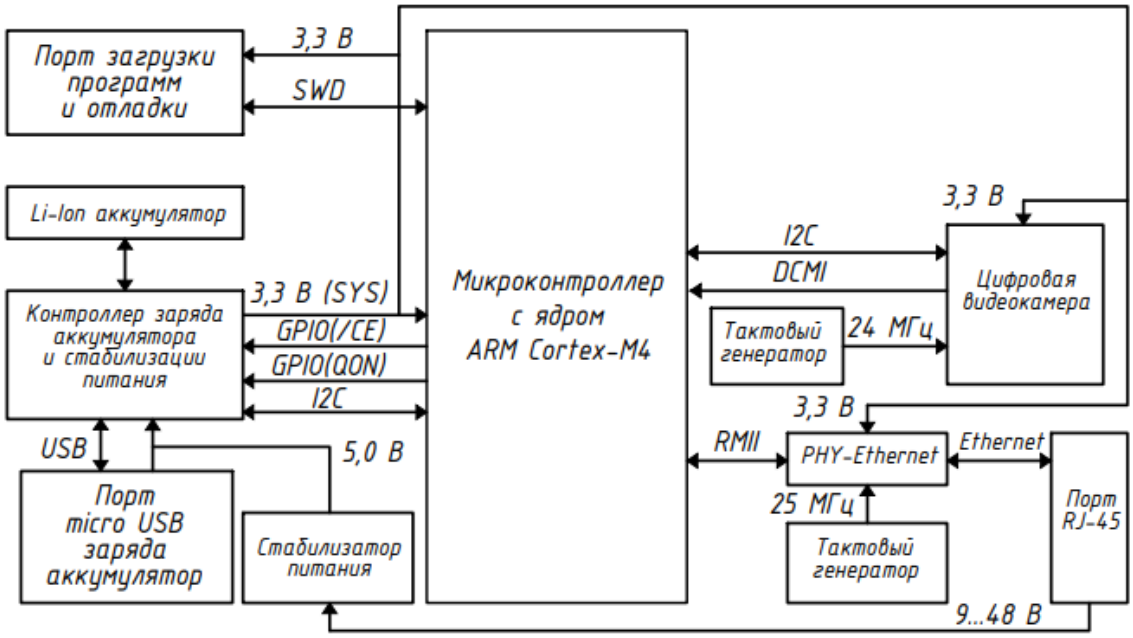



Рисунок 15 – Схема электрическая структурная

В следующих главах будет подробнее рассмотрена структурная схема мобильной *IP*-видеокамеры.

## 2.1 Обоснование базовых блоков структурной схемы *IP*–видеокамеры

Основными составляющими структурной электрической схемы являются следующие:

1 Микроконтроллер с ядром *ARM Cortex-M4* является одним из самых важных компонентов устройства, т.к. на его плечи ляжет ответственность за логику работы устройства (обработка сигнала, сжатие, связь с сервером и т.д.).

2 Блок цифровой видеокамеры будет представлен модулем камеры OV9655, о котором подробно рассказано в одной из глав выше. Основное назначение данного блока – получение цифрового сигнала или набора бит, который определяет какое-либо изображение и отправка этого набора в микроконтроллер.

3 Тактовый генератор видеокамеры нужен, чтобы синхронизировать сигналы, идущие от камеры к контроллеру. Может быть как внешним, так и встроенным в микроконтроллер.

4 Ещё один тактовый генератор для питания модуля *Ethernet.*

5 Непосредственно Ethernet модуль в виде микросхемы *LAN8720*, о которой подробно написано в одной из глав выше. Задача данного блока – взаимодействие микроконтроллера с подключаемым устройством, например, ноутбуком, через порт *RJ*-45, передача пакетов данных между микроконтроллером (то есть самим устройством-видеокамерой) и устройством, подключенным по *RJ*-45.

6 Порт *RJ-45*, через который будет осуществляться подключение каких-либо прочих устройств, например, ноутбука, с целью воспроизведения картинки с модуля видеокамеры.

7 Блок загрузки и отладки программ, встроенный в микроконтроллеры серии *STM32*, нужен, чтобы прошивать микроконтроллер, а также проводить отладку запущенных там программ. Взаимодействие осуществляется через интерфейс *SWD.*

8 Блок стабилизатора питания нужен для зарядки аккумулятора или питания устройства от порта *RJ-45*, благодаря технологии *PoE*, принцип работы которой описан в одной из глав выше.

9 Непосредственно литий-ионный аккумулятор, от которого можно некоторое время питать устройство.

10 Порт *Micro USB*, через который будет осуществляться питание устройства и зарядка аккумулятора.

11 Контроллер заряда и стабилизатор питания – микросхема, которая осуществляет питание микроконтроллера, контроль заряда аккумулятора и

функцию стабилизации питания при питании от *PoE*. Может быть представлена в виде микросхем *LTC4058* или *Bq24295,* описание которых представлено выше.

В следующей главе будут описаны связи (линии подключения), через которые происходит взаимодействие элементов проектируемого устройства.

## 2.2 Обоснование связей структурной схемы *IP*–видеокамеры

Микроконтроллер в данной схеме, как и в устройстве в целом, является одним из самых важных элементов устройства, поэтому все подключения основные так или иначе связаны с микроконтроллером.

Отладка и прошивка микроконтроллера происходит через интерфейс *SWD*. *SWD* расшифровывается как *Serial Wire Debug,* это более современная версия интерфейса *JTAG*, требующая для работы только 2 сигнальных выводов вместо как минимум 4 у стандартного *JTAG*.

Линии питания обеспечивают питание элементов устройства. В основном используют 3.3 В, однако некоторые пины могут выдерживать и 5В. Технология *PoE* способна выдавать от 9 до 48 В, что очень много для питания элементов, поэтому стабилизатор напряжения будет преобразовывать данный диапазон в стабильные 5 В, которые будут идти в контроллер заряда и дальше преобразовываться в 3.3 В.

*DCMI (Digital camera interface)* используется для подключения камеры с параллельным интерфейсом к микроконтроллеру. Камера формирует параллельный поток данных: сигналы *DCMI_D*[0…13] и сигнал тактирования *DCMI_PIXCLK*.

*RMII* расшифровывается как *Reduced Media Independent Interface* – сокращенный независящий от среды передачи интерфейс. Представляет собой интерфейс использующий сокращенный набор сигналов интерфейса *MII* и применяется для подключения *MAC*-блока *Ethernet* к блоку *PHY*.

*I2C* шина является одной из модификаций последовательных протоколов обмена данных. В стандартном режиме обеспечивается передача последовательных 8-битных данных со скоростью до 100 кбит/с, и до 400 кбит/с в "быстром" режиме. Для осуществления процесса обмена информацией по I2C шине, используется две линии: линия данных *SDA* и линия синхронизации *SCL*

# 3 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ IP–ВИДЕОКАМЕРЫ

## 3.1 Обоснование выбора САПР для разработки принципиальной электрической схемы

На данный момент существует множество систем автоматизированного проектирования для разработки схем для различных электрических устройств, ниже приведены некоторые из них [33]:

- *Altium Designer*;
- *Inventor*;
- *SOLIDWORKS*;
- *SketchUp*;
- *Autodesk EAGLE*;
- *KiCad EDA*.

Лидером из этого списка является САПР *Altium Designer*. Ниже приведены причины выбрать *AD* вместо других САПР:

1 *AD* – это единая платформа для проектирования. Это не просто программа с узкой специализацией, а полноценная рабочая среда инженера-проектировщика. Здесь можно создать плату электронного устройства от «а» до «я». Для каждого этапа разработки предусмотрены различные редакторы. Функционал позволяет создать схему, перенести с нее данные, сделать контур платы, создать *3D*-модель, задать правила проектирования и даже сформировать полный пакет выходных документов.

2 Интуитивно понятный интерфейс и удобный дизайн. Здесь нет лишних кнопок и диалоговых окон, рабочую область легко настроить под себя. Цветовая схема (светлый текст на темном фоне) хорошо воспринимается и не отвлекает от проектирования многослойных печатных плат.

3 Комфортная работа с библиотеками. Создание компонента, как правило, трудоемкий процесс, но в данной САПР, по заверению разработчика, все проще. Можно создать с нуля свою библиотеку компонентов и моделей или загрузить готовую. Есть несколько видов библиотек – для символов, посадочных мест, трехмерных моделей и текстовых *SPICE*. Компоненты библиотеки содержат все необходимые свойства для решения конструкторских задач на каждой стадии.

4 Интерактивная трассировка. Соединять компоненты между собой помогает высокоэффективное ядро. Разводка платы происходит одним кликом мыши, также есть автотрассировщик и ручная разводка трасс.

5 Экспорт и импорт файлов из других САПР. Программа позволяет открывать проекты, созданные в любой популярной САПР. Например, проекты платы из *AD* отображаются в *AutoCAD* или *SolidWorks* без потери качества.

6 *3D*-моделирование. С помощью встроенного *3D*-модуля можно увидеть плату в трехмерном представлении. Визуализация помогает обнаружить ошибки конструкции и мгновенно исправить их перед прототипированием.

7 Многопользовательский доступ и система контроля версий. Для этого в *AD* встроен инструмент *Git*.

Таким образом, можно сказать, что систем проектирования много, но именно в *Altium Designer* реально создать печатную плату любой сложности за короткий срок, не прибегая к дополнительным сервисам, плагинам и инструментам, так как интерфейс программы удобен и его одновременно достаточно для полноценного проектирования.

## 3.2 Описание используемых библиотечных элементов и процесса их создания

В соответствии со структурной схемой, на принципиальной схеме должны быть следующие элементы:

– микроконтроллер;
– модуль *PHY-Etnernet*;
– модуль видеокамеры;
– порт *RJ-45*;
– тактовый генератор 24 МГц;
– тактовый генератор 25 МГц;
– порт загрузки и отладки;
– стабилизатор питания;
– аккумулятор;
– порт *Micro-USB*;
– контроллер заряда и питания.

В целях оптимизации времени выполнения принципиальной схемы, было решено использовать стороннюю утилиту для *Altium Designer*, которая называется *ALTIUM Loader* (рисунок 16).

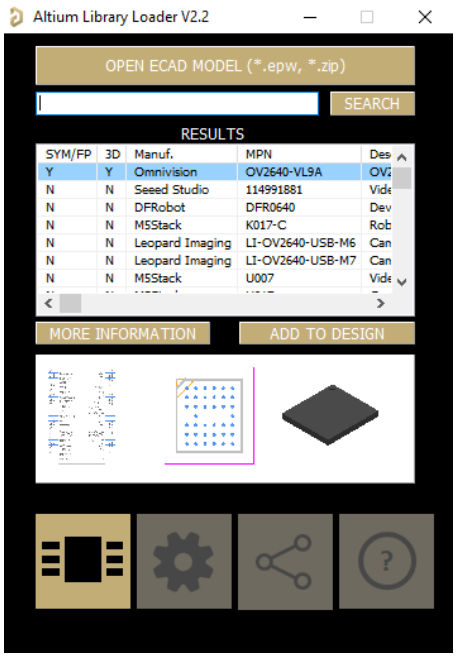



Рисунок 16 – Утилита *Altium Loader*

Данная утилита нужна для того, чтобы загружать библиотеки с различными компонентами с сайтов производителей этих компонентов. Например, таким образом была импортирована библиотека с микроконтроллером *STM32F407GT6* и модулем *PHY-Etnerhet LAN8720*. Однако, только что импортированный компонент *LAN8720* выглядит следующим образом (рисунок 17):

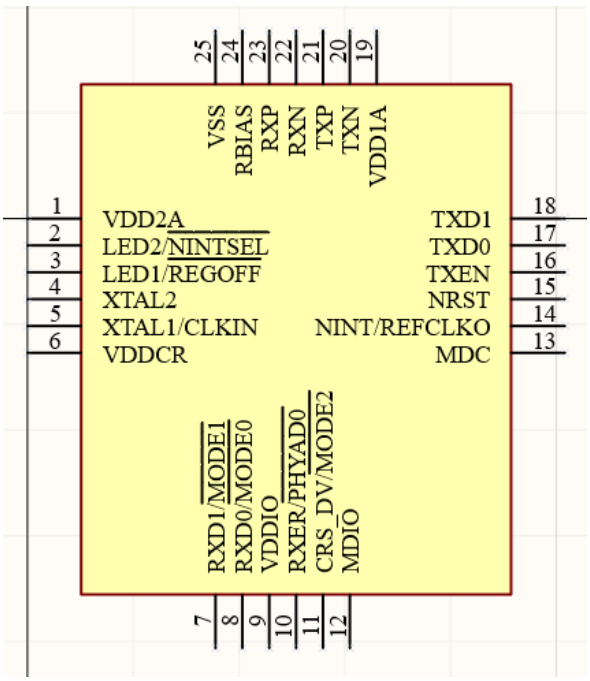



Рисунок 17 – Пример импортированного компонента

Как можно заметить, графическое исполнение данного компонента выполнена не по ГОСТу. Поэтому, каждый импортируемый элемент был перерисован в соответствии с ГОСТ 2.743-91. Пример перерисованной схемы *LAN8720* представлен на рисунке 18.

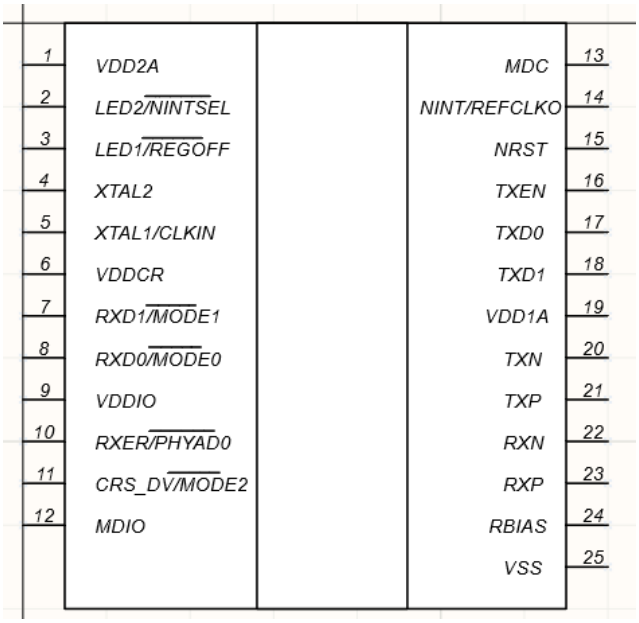



Рисунок 18 – Перерисованный импортированный компонент

Таким образом были созданы основные компоненты схемы.

### 3.3 Обоснование выбора базовых компонентов принципиальной схемы *IP*–видеокамеры

Что касается выбора компонентов, выбор был произведён согласно заданию. Среди микроконтроллеров с ядром *ARM Cortex-M4* компания *STM* предлагает микроконтроллеры серии *STM32F407/417*. Серия 417 отличается наличием шифрующих модулей, в остальном – от рабочей частоты до рабочей температуры одинаково. Таким образом, была выбрана 407 серия. Что касается модели (рисунок 19), то их отличают 2 буквы в конце названия: первая *V* означает, что микроконтроллер имеет 100 пинов, *Z* – 144, *I* – 176, а вторая – объём оперативной памяти: *E* – 512 КБ, *G* – 1 МБ. Исходя из всего этого, был выбран микроконтроллер *STM32F407VG*.

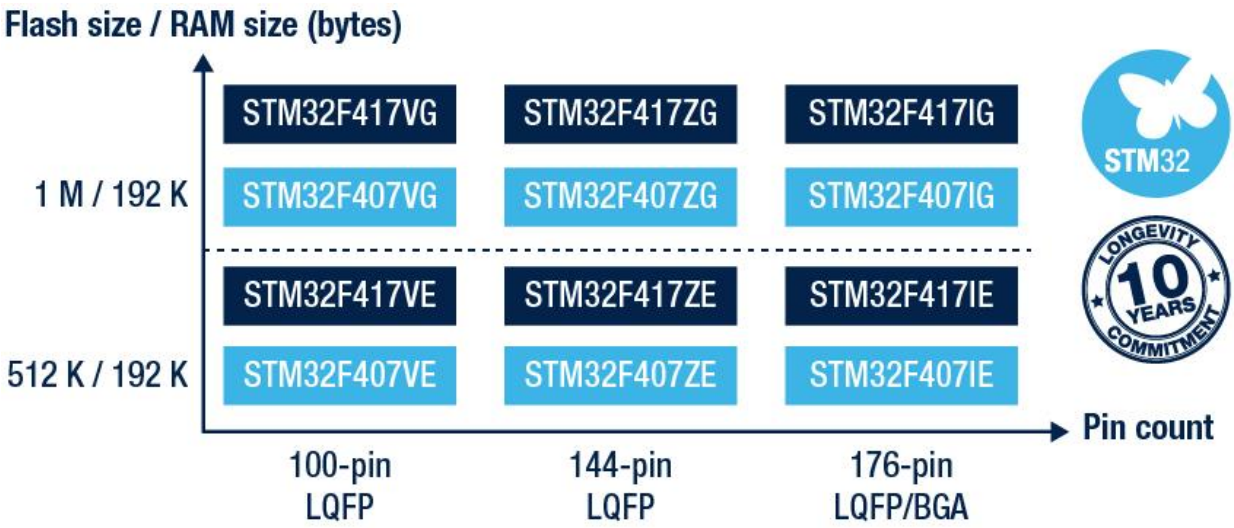



Рисунок 19 – *STM32F407/417*

Далее по списку – модуль *PHY Ethernet*. В качестве данного модуля была выбрана уже описанная ранее микросхема *LAN8720A*. В дополнение к ней идёт частотный генератор *405C11A25M00000,* который должен генерировать частоту 25 МГц. Также необходимо было добавить непосредственно порт *RJ-*45 в виде компонента *SS*-6488, а именно модульного коннектора с интерфейсом *RJ-45*.

Модуль видеокамеры был выбран также согласно заданию – *OV9655(2640).* Модель 9655 и 2640 идентичны по своей сути (рисунок 20) и имеют одинаковые принципиальные схемы на сайте производителя, поэтому не имеет разницы, какой модуль использовать на принципиальной схеме проектируемого устройства.


Рисунок 20 – *OV9655(2640)*

Как и в случае с *Ethernet* модулем, модулю видеокамеры также нужен генератор частот. В качестве такового был выбран *405C11A24M00000* с частотой 24 МГц.

Интерфейс отладки нужен для прошивки контроллера и его дальнейшей отладки, и в качестве такого интерфейса был создан схематический вид *SWD*-коннектора, а также импортировано посадочное место.

Следующие компоненты относятся к питанию устройства.

В качестве контроллера заряда аккумулятора и стабилизации питания была выбрана описанная ранее схема *BQ24295.* Питание будет распределяться данной микросхемой и поступать через *PoE* или через *Micro-USB.* В первом случае поступающее напряжение может быть нестабильно, что можно нанести вред устройству, поэтому нужно было добавить стабилизатор питания – таким стал *AP2202K-ADJTRG1*. Что касается *Micro-USB,* то используется модуль *(micro-USB)0473461001.*

### 3.4 Обоснование связей принципиальной электрической схемы *IP*–видеокамеры

Для корректной работы устройства необходимо правильно расставить связи между компонентами. Расстановка производилась согласно технической документации и принципиальным схемам выбранных элементов.

Связи модуля видеокамеры:

– *E3 (Pixel CLocK output),* выполняет роль генератора частоты для синхронизации с периферийными устройствами и должен быть подключён к выходу *PA6 (DCMI_PCLK*) микроконтроллера;

– *E5(Y6), F3(Y2), F4(Y4), F5(Y8), G3(Y3), G4(Y5), G5(Y7), G6(Y9)* нужны для передачи данных в формате 8-*bit RGB* и подключаются к выходам *PE4(DCMI_D4), PE0(DCMI_D2), PC6(DCMI_D0), PE5(DCMI_D6), PE1(DCMI_D3), PC7(DCMI_D1), PB6(DCMI_D5), PE6(DCMI_D7)* микроконтроллера соответственно;

– *D2(VSYNC),* нужен для вертикальной синхронизации изображения и подключается к выходу *PB7(DCMI_VSYNC)* микроконтроллера;

– *C4(XVCLK),* обеспечивает тактирование датчика изображения внутри модуля видеокамеры, подключается к внешнему тактовому генератору частотой 24 МГц;

– *C1*(*SIO_D),* является аналогом и выполняет роль линии данных интерфейса *I2C* и подключается к выходу *PB9(I2C1_SDA)* микроконтроллера;

– *C2*(*SIO_C),* является аналогом и выполняет роль линии тактирования интерфейса *I2C* и подключается к выходу *PB8(I2C1_SCL)* микроконтроллера;

– *C3*(*HREF),* нужен для горизонтальной синхронизации изображения и подключается к выходу *PA4(DCMI_HSYNC)* микроконтроллера;

Линии *DOGND, AGND, DOVDD, AVDD, PWDN* подключались согласно принципиальной схеме *OV9655.*

Модуль коннектора *RJ-45* соединяется с модулем *PHY-Ethernet* для передачи данных, а также с модулем стабилизации питания *AP2202K* для обеспечения питания по технологии *PoE.* Связи модуля *RJ-45*:

– *TD+(transmit positive), TD- (transmit negative), RD+(receive positive), RD- (receive-negative),* нужны для передачи электрических сигналов и соединяются со входами *TXP, TXN, RXP, RXN* схемы *LAN8720ACP* соответственно, согласно документации;

– *TCT, RCT, MH1, MH2,* нужны для питания по технологии *PoE* (то есть, соединяются с выходами схемы *AP2202*), а также, согласно документации, должны быть заземлены.

Связи схемы *PHY-Ethernet LAN8720ACP:*

– *MDC,* работает в качестве линии данных интерфейса *RMII* и подключается к выходу *PA2(ETH_MDIO)* микроконтроллера;

– *REFCLKO,* используется для генерации опорного тактового сигнала и подключается к выходу *PA1(ETH_RMII_REF_CLK)* микроконтроллера, а также к генератору частоты 25 МГц;

– *NRST,* отвечает за сброс микросхемы, подключается к выходу *PD3* микроконтроллера;

– *TXEN,* определяет процесс передачи данных, подключается к выходу *PB11(ETH_RMII_TX_EN)* микроконтроллера;

– *TXD0, TXD1,* используются для передачи данных на приёмопередатчик, подключаются к выходам *PB12(ETH_RMII_TXD0)* и *PB13(ETH_RMII_TXD1)* микроконтроллера соответственно;

– *RBIAS,* который выводится на землю через резистор сопротивлением 12.1 кОм;

– *MDIO,* работает в качестве тактирующей линии интерфейса *RMII* и подключается к выходу *PC1(ETH_MDC)* микроконтроллера;

– *CRS_DV,* используется, чтобы определять состояние среды передачи (свободна или простаивает), подключается к выходу *PA7(ETH_RMII_CRS_DV)* микроконтроллера;

– *RXD0, RXD1,* используются для приёма данных приёмопередатчиком, подключаются к выходам *PB12(ETH_RMII_TXD0)* и *PB13(ETH_RMII_TXD1)* микроконтроллера соответственно;

– *XTAL1, XTAL2,* являются входом и выходом для внешнего кварцевого резонатора соответственно, подключаются согласно документации;

– *VDDIO, VDDCR, VDD1A, VDD2A,* отвечают за питание схемы, подключаются согласно документации;

– *LED1, LED2,* служат для определения режима работы схемы, подключаются согласно документации.

Связи стабилизатор питания *AP2202K*:

– *VIN*, (входное напряжение), *EN* (вход включения), подключаются к шине;

– *ADJ/BYP* (регулировка выхода), *GND* (земля), выводятся на землю;

– *VOUT* (выходное напряжение), выводится к месту, где заземляется коннектор *RJ-45;*

Связи модуля отладочного интерфейса:

– *RESET,* отвечает за сброс состояния контроллера, подключается к выходу *NRST* микроконтроллера;

– *SWCLK,* выступает в роли тактирующей линии *I2C*, подключается к выходу *PA14(JTCK-SWCLK)* микроконтроллера;

– *SWDIO,* выступает в роли линии данных *I2C*, подключается к выходу *PA13(JTMS-SWDIO)* микроконтроллера.

В следующей главе будет описана схема зарядки проектируемого устройства.

### 3.5 Анализ и обоснование принципиальной электрической схемы зарядки аккумуляторной батареи

В качестве электрической схемы зарядки аккумуляторной батареи будет выступать описанная ранее схема *BQ24295*. Принципиальная схема *BQ24295* была приведена на рисунке 14, соответственно, при проектировании принципиальной схемы видеокамеры, схема *BQ24295* была частично продублирована.

*BQ24295* выполняет следующие функции:

– переключение режимов зарядки с эффективностью 90 %;

– зарядка от *USB* с напряжением от 3,9 В до 6,2 В с защитой от перенапряжения 6,4 В;

– определение *USB*-хоста и зарядных портов *D+/D-;*

– поддержка входного напряжения и тока стандартов USB2.0 и USB 3.0;

– ограничение входного тока: 100 мА, 150 мА, 500 мА, 900 мА, 1А, 1,5А, 2А и 3А;

– регулируемый выход 4,55В – 5,5В при 1,5А, КПД 90% в разогнанном режиме;

– мгновенное включение системы без батареи или с глубоко разряженной батареей;

– порт *I2C* для оптимальной производительности системы и отчетности о состоянии системы;

– автономная зарядка аккумулятора с управлением или без управления хостом;

– прекращение зарядки и защита от перезарядки;

– регулировка напряжения заряда ±0,5%;

– регулировка зарядного тока ±7%;

– регулировка входного тока ±7,5%;

– стабилизация выходного напряжения ±3% в форсированном режиме;

– управление питанием;

– синхронное переключение МОП-транзисторов;

– определение температуры аккумулятора для зарядки и разрядки как в разогнанном режиме, так и в обычном;

– таймер безопасности зарядки аккумулятора;

– терморегуляция и тепловое отключение;

– защита от перенапряжения на входе и в системе;

– MOSFET Защита от перегрузки по току;

– возможность отслеживания максимальной мощности по входу;

– регулировка напряжения.

Таким образом, сочетая в себе все перечисленные возможности, а также малые размеры (4 на 4 миллиметра), данная схема зарядки была включена в состав проектируемого устройства.

Питание будет осуществляться через порт *Micro-USB,* связи которого было установлены согласно принципиальной схеме выбранного модуля.

Что касается технологии *Power over Ethernet,* то в целях предотвращения перенапряжения и вывода устройства из строя, был установлен стабилизатор напряжения *AP2202K-ADJ*, который был подключён согласно своей принципиальной схеме.

# 4 РАЗРАБОТКА ПО И ПРОГРАММИРОВАНИЕ АЛГОРИТМА ФУНКЦИОНИРОВАНИЯ *IP*-ВИДЕОКАМЕРЫ В СРЕДЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

## 4.1 Обоснование источника потока видео образов

Основной рассматриваемой темой данной главы будет передача потока видео образов с вебкамеры, обработка потока, отображение и сохранение.

Вебкамера – это устройство захвата видео, не имеющее собственного процессора и сетевого интерфейса. Вебкамера требует подключения к компьютеру, смартфону, либо другому устройству, имеющему сетевую карту и процессор. Обычная камера, как правило сжимает видео в *H.264* кодек и может работать в двух режимах транспортировки данных: *interleaved* и *non-interleaved.* Для того чтобы забрать видео с камеры с минимальной задержкой, нужно использовать non-interleave mode и получать видеотрафик по *UDP*.

В качестве языка программирования для разработки ПО был выбран *JavaScript*, основной причиной тому было то, что данный язык поддерживается всеми современными браузерами, как десктопными, так и мобильными, таким образом обеспечивается кроссплатформенность.

Браузеры не поддерживают стек протоколов *RTSP / UDP* напрямую, но поддерживают стек протоколов встроенной технологии *WebRTC*.

*JS* может работать с множество полезных программных интерфейсов (*API)*. Например, практически везде используется *API* для работы с документами, загруженными в браузер [34]. Явный пример – *DOM (Document Object Model) API*, позволяющий работать с *HTML* и *CSS* – создавать, удалять и изменять *HTML*, динамически изменять вид страницы и т.д. Другое категорией *API* является аудио и видео *API*, такие как *HTMLMediaElement, Web Audio API*, и *WebRTC*, которые позволяют проводить различные операции с мультимедиа. Например, создать собственный пользовательский интерфейс для проигрывания аудио/видео, вывод на экран субтитров, записывать видео с веб-камеры для дальнейшей обработки или для передачи на другой компьютер или сервер, применять звуковые эффекты к аудиофайлам, видеофайлам и т.д.

В данной работе используется *MediaRecorder WebAPI,* который является интерфейсом *MediaStream Recording API.* В свою очередь *MediaStream Recording API* делает возможным данные, сгенерированные в *MediaStream* или *HTMLMediaElement* объектом для анализа, обработки или сохранения в дисковое пространство устройства. *MediaStream Recording API* в свою очередь связан с интерфейсом *WebRTC* [35].

*WebRTC (Web Real-Time Communications)* – это технология, которая позволяет *Web*-приложениям захватывать и выборочно передавать аудио и/или видео медиа-потоки, а также обмениваться произвольными данными между браузерами, без обязательного использования посредников.

Набор стандартов, которые включает в себя технология *WebRTC*, позволяет обмениваться данными и проводить пиринговые телеконференции, без необходимости пользователю устанавливать плагины или любое другое стороннее программное обеспечение.

Таким образом, данное веб приложение предоставляет сервис, который использует любое встроенное или подключаемое устройство генерации мультимедиа.

### 4.2 Разработка диаграммы состояний *IP*–видеокамеры

Описать поведение отдельно взятого объекта помогает диаграмма состояний. Диаграмма состояний показывает все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

На этапе включения происходит инициализация периферийных устройств, которая включает в себя несколько этапов:

– определение структуры периферии;

– включение тактового генератора для периферии

– запрос базовой конфигурации устройств;

– запрос дополнительной конфигурации устройств;

– включение периферии;

На этом этапе должна произойти инициализация основных периферийных модулей, на диаграмме состояний это модуль камеры и модуль *Ethernet.*

Следующий этап – захват изображения с камеры. Данный процесс происходит всё время, пока работает устройство. Одновременно происходит обработка полученных изображений. На диаграмме обработка изображения была объединена в один этап. Более подробное описание процесса обработки изображения будет описано в главе 4.3 и 4.6.

После обработки изображения, полученный в результате обработки массив бит выгружается в память. Далее эти данные, уже в сжатом виде, передаются в сеть с помощью *Ethernet.*

### 4.3 Разработка схемы алгоритма функционирования *IP*-видеокамеры

Диаграмма состояний показывает, в каких состояниях находится объект во время цикла работы. В свою очередь схема алгоритма функционирования показывает, по какому принципу происходят переходы к тому или иному состоянию, а также операции в каждом состоянии.

На этапе включения устройства происходит инициализация периферийных модулей, происходит установка тех или иных регистров. На

данном этапе происходит инициализация *GPIO, I2C, RMII, DCMI,* а также инициализация модуля камеры. Алгоритм инициализации, как правило, выглядит так:

– включается тактирование порта;

– устанавливается скорость порта;

– устанавливается тип порта;

– устанавливается, куда подтянут вывод;

– указывается, чем порт будет заниматься.

– если порт подключен к какому-либо устройству, задаётся альтернативная функция.

Начиная с этого момента камера начинает передавать непрерывный поток изображений. После того, как изображение получено процессором, оно начинает обрабатываться. Данный процесс можно разбить на 6 стадий:

– разбиение изображения на макро-блоки;

– прогнозирование;

– преобразование;

– квантование;

– кодирование;

– форматирование в поток бит.

Следующий шаг в алгоритме – отправка полученных данных по *Ethernet*. Для этого *PHY* модуль забирает уже сжатое изображение с оперативной памяти, кодирует его и отправляет по адресу подключенного устройства. Если по каким-то причинам отправка невозможна, кадр сбрасывается и цикл принятия изображения повторяется. Данный процесс выполняется, пока устройство не будет выключено.

### 4.4 Обработка и передача потока видео образов

Как уже было сказано, в приложении видеопоток захватывается через *MediaRecorder*. Отличительной чертой браузерного воспроизведения медиа является поддерживаемый формат. В различных браузерах (*Edge, Chrome, Firefox)*, а также на различных операционных системах (*iOS, Android)* поддерживаются различные аудио и видео кодеки. Логика работы приложения была построена таким образом, чтобы воспроизведение и запись видеопотока было возможно на любом устройстве.

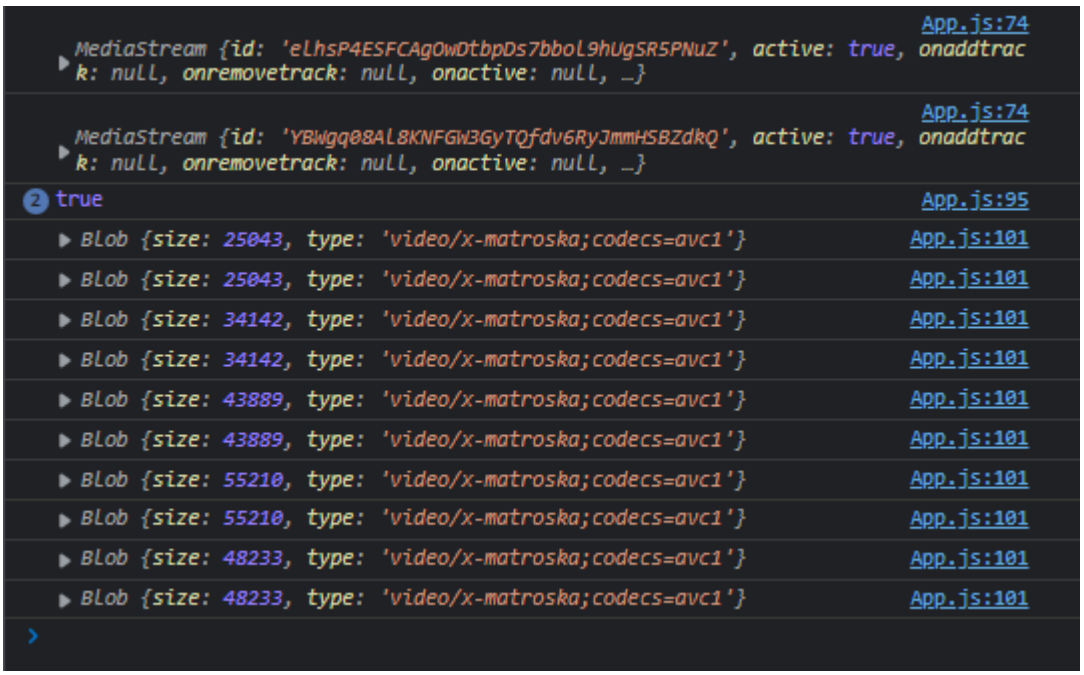



Рисунок 21 – Обработка потока данных

Как можно заметить на рисунке выше, в каждый момент, когда камера передаёт изображение, это изображение не только воспроизводится на экране пользователя, но и дублируется в отдельный канал в виде потока байт. Данный канал может быть использован, чтобы, например, пересылать изображения на сервер, или для дальнейшей обработки изображения.

### 4.5 Реализация пользовательского интерфейса

Интерфейс приложения был создан с помощью библиотеки *JavaScript – React*. В случае браузерного интерфейса, контейнером для объектовявляется модель объекта документа (*DOM*). Виртуальный *DOM* является представлением реального *DOM*, который строится и манипулируется браузерами. Усовершенствованные библиотеки, такие как *React*, генерируют дерево элементов в памяти, эквивалентное реальному *DOM*, который формирует виртуальный *DOM* декларативным образом.

Таким образом, используя *React*, можно совмещать создание логики приложения и его интерфейса.

Интерфейс приложения представлен несколькими блоками:

– блок кнопок;

– блок, где выводится изображение с камеры;

– блок с диаграммой, которая показывает работу кодека;

– блок с дополнительной информацией.

Блок кнопок представлен четырьмя кнопками. Кнопка *Start Camera* запрашивает разрешение на использование аудио и видео модулей. Кнопки

*Start Record* и *Stop Record* начинают и, соответственно, останавливают запись трансляции, генерацию диаграммы и дополнительной информации.

Блок с изображением камеры имеет адаптивный размер, в зависимости от экрана устройства и от возможностей веб-камеры.

Блок с диаграммой выглядит следующим образом (рисунок 22):


Рисунок 22 – Блок с диаграммой

В данном блоке отображается размер захваченного кадра через каждую секунду. Данная диаграмма отлично показывает такие методы, как прогнозирование и преобразование, которые используются кодеком для сжатия видеопотока.

### 4.6 Алгоритм реализации стандарта сжатия *H*.264 в контексте *LWIP*–стека

Как было сказано ранее, *LWIP* – это небольшая реализация стека протоколов *TCP/IP*, целью создания которого было сокращение использования ресурсов системы и сохранение возможностей полномасштабного стека протоколов *TCP/IP*. Это делает возможным применение *LWIP* во встраиваемых системах, где существуют жёсткие ограничения как в оперативной, так и внутренней памяти.

Видеопоток, который передаётся по сети, обладает таким важным параметром, как битрейт. Битрейт – это общее количество битов, которое может передаваться в единицу времени. Обычно измеряется в секундах и выражается как килобиты в секунду, мегабиты в секунду или гигабиты в

секунду. Битрейт *IP* -камеры напрямую влияет на использование полосы пропускания в сети. Пропускная способность – это термин, используемый для описания максимального объема данных, которые могут перемещаться вдоль определенного канала за фиксированное количество времени.

Пропускная способность – это, вероятно, самая главная составляющая сети. Если система наблюдения использует больше пропускной способности, чем доступно, то могут возникнуть непредвиденные перерывы в работе системы.

*H*.264 снижает битрейт видео для снижения использования полосы пропускания. Снижение битрейта означает, что может быть передано больше данных, соответственно, можно повысить качество видео без ущерба пропускной способности сети. Низкий битрейт с потерями качества изображения обеспечен различными методиками сжатия.

Итак, видео без предварительной обработки – это последовательность двумерных массивов, содержащих информацию о пикселях каждого кадра. Таким образом это трёхмерный (2 пространственных измерения и 1 временной) массив байтов. Каждый пиксель кодируется тремя байтами – один для каждого из трёх основных цветов (красный, зелёный и синий). Таким образом, если имеется видео в разрешении 1920 на 1080 пикселей, имеет частоту 60 кадров в секунду, это получается примерно 400 МБ/с данных.

Одним из методов сжатия видео является сжатие с потерями [37], где менее значимые части отбрасываются, а важные сохраняются. Таким образом, преобразуется только представление данных, а информационная энтропия – нет. Информационной энтропией называется минимальное количество единиц, которое необходимо для представления некоторых данных. Таким образом, энтропийное кодирование может подходить для кодировки любых данных.

Также при кодировании информации используется переход между разными системами координат. Например, данные, которые изменяются во времени (яркость пикселя, например), можно представить через частотную систему координат, и будет произведён переход к частотному пространству. Теорема Котельникова гласит, что при достаточно высокой частоте, любые данные можно представлять без потерь при любой степени сжатия. Здесь вводятся координаты *freqX и freqY*, которые являются базисом в частотной системе координат. При переходе между системами координат, в данном случае, используется преобразование Фурье.

Преобразование Фурье является важным инструментом обработки изображений, который используется для разложения изображения на синусоидальные и косинусные компоненты. Выходные данные преобразования представляют изображение в частотной области, в то время как входное изображение является эквивалентом пространственной области.

В изображении домена Фурье каждая точка представляет определенную частоту, содержащуюся в пространственном изображении домена.

Поскольку речь идет только о цифровых изображениях, имеет смысл ограничиться дискретным преобразованием Фурье (*DFT*) [38].

*DFT* является выборочным преобразованием Фурье и, следовательно, содержит не все частоты, образующие изображение, а только набор образцов, который достаточно велик, чтобы полностью описать пространственное доменное изображение. Количество частот соответствует количеству пикселей в пространственной области изображения, т.е. изображение в пространственной области и область Фурье имеют одинаковый размер.

Для квадратного изображения размера *N*N* двумерное *DFT* задается формулой 1:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{i=0}^{N-1} f(i,j) e^{-i2\Pi\left(\frac{ki}{N}+\frac{lj}{N}\right)} \quad (1)$$

Данная формула должна быть применена к каждой точке изображения, и, так как преобразование Фурье отделимо, данную сумму можно разбить на две независимые суммы. Используя формулу 1, пространственное доменное изображение сначала преобразуется в промежуточное изображение с помощью *N* одномерных преобразований Фурье. Это промежуточное изображение затем преобразуется в конечное изображение, опять же с использованием *N* одномерных преобразований Фурье. Выражение двумерного преобразования Фурье в терминах ряда одномерных преобразований 2N уменьшает количество требуемых вычислений. Чтобы ещё больше увеличить скорость вычислений, используется быстрое преобразование Фурье.

Результат такого преобразования представлен на рисунке 23.

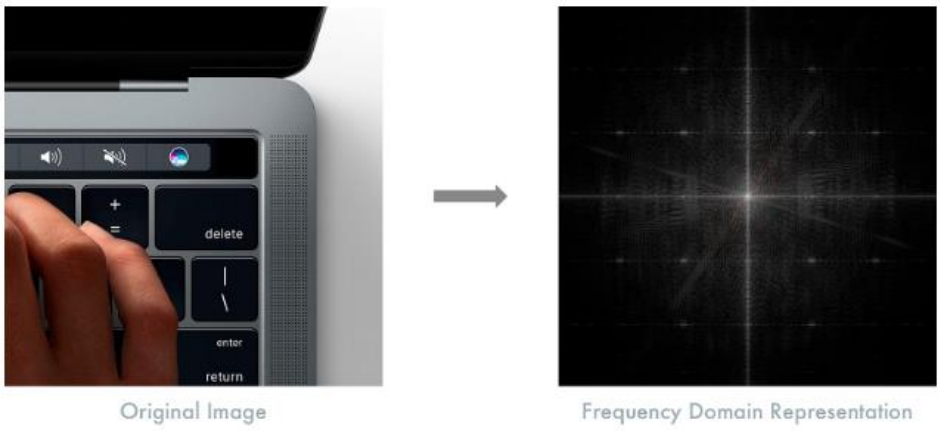



Рисунок 23 – Результат преобразования Фурье

Мелкие детали имеют более высокую частоту и, соответственно, более плавные изменения. Далее, двигаясь от краёв диаграммы к центру, можно регулировать детализацию фото, и, соответственно, размер.

Ещё одним методом сжатия изображения является цветовая обработка. Технология кодирования изображения со снижением цветового разрешения – это цветовая субдискретизация. Данная технология основана на том, что человеческий глаз менее чувствителен к перепадам цвета, чем к перепадам яркости. Таким образом можно отсекать плохо различимые части изображения без относительной потери качества, однако данный подход работает по-разному относительно каждого человека.

Последним рассматриваемым методом сжатия является компенсация движения [39].При такой технологии кодируются только различия векторов движения, тем самым обеспечивая высокую степень сжатия любого видео с перемещениями.

Первым шагом является разделение кадра на несколько разделов, подразделов и далее. При дроблении изображения можно точнее прогнозировать вектор движения, используя как можно меньшие блоки для движущихся частей, и более крупные блоки для статичных частей. Обычно кодеки организуют эти разделы во фрагменты, макроблоки и множество подразделов. Максимальный размер этих разделов варьируется, HEVC устанавливает 64x64, в то время как AVC использует 16x16, а подразделы могут дробиться до размеров 4x4.

Следующим шагом является прогнозирование. Когда уже созданы разделы, передаются векторы движения и остаток, направление прогноза и остаток для *INTER*-прогнозирования и *INTRA*-прогнозирования соответственно.

Далее на основе остаточных блоков рассчитываются пиксели, которые можно отбросить. Это может делаться с помощью, например, дискретного косинусного преобразования:

– преобразование блоков пикселей в одинаковые по размеру блоки частотных коэффициентов;

– устранение пространственной избыточности;

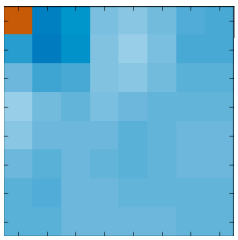
– обеспечение обратимости.

В качестве примера можно рассмотреть Рисунок 24 – Произвольный блок изображения, на котором представлен произвольный блок изображения и соответствующий ему массив пикселей.

```
[[227, 255, 255, 155, 146, 158, 169, 178],
 [247, 255, 255, 161, 171, 186, 194, 197],
 [255, 255, 246, 193, 198, 197, 194, 192],
 [226, 206, 193, 193, 192, 189, 189, 188],
 [191, 191, 190, 190, 190, 190, 189, 190],
 [191, 191, 190, 191, 190, 191, 191, 191],
 [191, 191, 192, 192, 193, 193, 193, 193],
 [192, 193, 193, 195, 195, 195, 196, 197]]
```

Рисунок 24 – Произвольный блок изображения

После применения дискретного косинусного преобразования результат будет выглядеть следующим образом (рисунок 25):



```
[[ 1582. ,    87.2,    51.8,   -15.9,   -34.2,   -12.8,    14. ,    17.4],
 [   38.5,    96.1,    56. ,   -24.7,   -44.4,   -15.5,    17.1,    20.9],
 [   -3.2,    24.8,    20.3,   -26.2,   -28.9,    -9. ,     8.6,     9.8],
 [  -47.1,    -9.9,    -1.5,   -15.9,    -8.3,    -1.2,     0.9,    -1.4],
 [  -33. ,    -8.3,    -5. ,    -3. ,     7.8,     3.5,    -4.2,    -6. ],
 [   -8.3,     8.5,    -3.5,     2.3,     7.9,     2.9,    -3.7,    -4. ],
 [    8.6,    12.1,    -5.1,    -3.3,    -0.5,    -1.5,    -2. ,    -1.8],
 [    7.4,     5.8,    -6.7,    -5.9,    -5. ,    -3.8,     0.4,     0.9]]
```

Рисунок 25 – Результат применения *DCT*

Как можно заметить, что первый коэффициент сильно отличается от всех остальных. Этот первый коэффициент известен как *DC*-коэффициент, представляющий все выборки во входном массиве, нечто похожее на среднее значение. Также данный блок коэффициентов отделяет компоненты с высокими частотами от компонентов с низкими частотами. Таким образом можно отбросить избыточные данные практически без потери качества изображения.

Таким образом, убирая (приравнивая к 0) наименее важные коэффициенты, можно убрать практически 70% данных, при незначительной потере качества. Стоит также учитывать, что каждый коэффициент формируется с использованием всех пикселей.

Следующий шаг – квантование. Квантование заключается в том, чтобы особым образом изменять коэффициенты так, чтобы изменения были обратимы, и чтобы ненужная информация терялась.

После квантования данные всё еще можно сжимать относительно без потерь, для этого применяют различные виды кодирования. Существуют различные виды кодирования, например, *VLC* кодирование, арифметическое кодирование. *H.264* использует энтропийное кодирование, которое было описано ранее.

Последний шаг метода компенсации движения – вывод в формат битового потока. Сжатые кадры распаковываются декодером, для этого декодеру должна быть предоставлена вся информация о процессе кодировки.

Таким образом, в данной главе был разобран алгоритм сжатия *H.264* и можно сделать вывод, что данный кодек является эффективным средством сжатия видео, так как требования отрасли видеонаблюдения постоянно стремятся к более качеству изображения без какого-либо компромисса по количеству кадров. С обычными форматами сжатия это было невозможно, но с внедрением *H*.264 можно увидеть много преимуществ, которые могут улучшить качество видеонаблюдения.

# 5 РАЗРАБОТКА КОНСТРУКЦИИ ПРОЕКТИРУЕМОГО ПРИБОРА

## 5.1 Выбор и обоснование элементной базы

В соответствии со принципиальной схемой, основными компонентами на плате являются:

– микроконтроллер *STM32F407VGT6*;
– модуль *PHY-Etnernet LAN8720*;
– модуль видеокамеры *OV9655*;
– порт *RJ-45 SS-6488-FLS*;
– тактовый генератор 24 МГц *403C11A24M00000*;
– тактовый генератор 25 МГц *403C11A25M00000*;
– порт загрузки и отладки *SWD*;
– стабилизатор питания *AP2202K-ADJTRG1*;
– порт *Micro-USB 0473461001*;
– контроллер заряда и питания *BQ24295CRG*.

Вторичными компонентами являются резисторы, конденсаторы, диоды и т.д. Выбор компонентов производился из существующих на интернет-площадках радиодеталей.

Все резисторы – *SMD*, предназначенные для поверхностного монтажа. Резисторы типа 0805 – бескорпусные толстопленочные резисторы, предназначены для работы в цепях постоянного, переменного и импульсного тока. Номинальная мощность – 0.128 Вт, номинальное сопротивление – от 1 Ом до 10 МОм, масса – 0.01г, размеры – 1.25мм на 2мм.

Меньшие резисторы, вероятно, имеют меньшую мощность и номинальное напряжение, однако также имеют меньшие паразитные свойства, например шум. Как правило, типоразмер 0603 уже подходит в большинстве случаев, 0805 – тем более.

Керамические конденсаторы типа 0805 имеют хорошие импульсные характеристики и малый уровень собственных шумов благодаря низкому импедансу на высоких частотах. Диапазон возможных емкостей: 0.3 пФ – 100 мкФ, масса – 0.01г, размеры – 1.25мм на 2мм.

## 5.2 Выбор и обоснование конструктивных элементов и установочных изделий

Основные конструктивные элементы печатной платы – основание (подложка) и проводники. Ко второстепенным элементам относят:

– контактные площадки;
– переходные металлизированные и монтажные отверстия;

– ламели для контактирования с разъемами, участки для осуществления теплоотвода и прочее.

В качестве исходного материала выступает диэлектрическое основание, ламинированное с одной или двух сторон медной фольгой. В качестве диэлектрика могут выступать:

– листы, изготовленные на основе стеклотканей, пропитанных связующим на основе эпоксидных смол – стеклотекстолит (СФ, СТФ, СТАП, *FR4* и т.п.);

– листы с керамическим наполнителем, армированные стекловолокном – *Rogers RO5603, RO4350;*

– листы фторопласта (*PTFE*), также армированные – *Arlon AD* 250 и 255, *Arlon* (*AD* и *AR*);

– ламинаты на металлическом основании (алюминий, медь, нержавеющая сталь);

– плёнки из полиимида, полиэтилентерефталата (PET, ПЭТФ, лавсан).

Проводники на печатной плате соответствуют классу точности 5, соответственно, проводники имеют ширину 0.1 мм. Расстояние между проводниками также равно 0.1 мм. Пятый класс точности печатных плат характеризуется высокой точностью и широко применим от производства бытовых приборов до вычислительной техники.

Контактная площадка – элемент печатного рисунка платы, используемый для дальнейшего выполнения паяного или иного соединения. В современных изделиях, чтоб повысить плотность монтажа, применяют вскрытые контактные площадки на внутренних слоях многослойных печатных плат. При этом во внешних слоях выполняются окна доступа.

Чтобы избежать перетекания припоя, произвольного смещения компонентов и других дефектов пайки, нельзя допускать расположения переходных отверстий на контактных площадках элементов или в непосредственной близости от них. Как уже говорилось, необходимо, чтобы контактные площадки компонентов были отделены от переходных отверстий, других контактных площадок и т.д. паяльной маской.

# 6 РАСЧЁТ КОНСТРУКТИВНО-ТЕХНОЛОГИЧЕСКИХ ПАРАМЕТРОВ ПРОЕКТИРУЕМОГО ПРИБОРА

## 6.1 Проектирование печатного модуля

### **6.1.1** Выбор типа конструкции печатной платы, класса точности и шага координатной сетки

Среди конструкций печатных плат можно выделить три типа: односторонние, двусторонние и многослойные печатные платы.

Достоинством конструкции односторонних печатных плат является простота и низкая стоимость изготовления. Недостатком является невозможность применения функционально сложных электронных компонентов ввиду низкой разрешающей способности печатных проводников. Ширина печатных проводников и соответственно расстояние между ними составляет около 1 мм и более, что соответствует первому классу точности в соответствии с ГОСТ Р 53429-2009. Кроме того, такую плату сложно реализовать с точки зрения обеспечения целостности сигнала, так как печатные проводники расположены на одной стороне, а также отсутствуют потенциальные слои, которые выполняют функцию экранировки. Более того, для реализации таких печатных плат используют в качестве материала гетинакс, что говорит об ограниченной области применения таких плат.

В зависимости от материала основания печатной платы различают двусторонние печатные платы на жестком или гибком основании. В отличие от односторонних печатных плат, двусторонние печатные платы находят более широкое распространение, так как путем применения данной конструкции удается решить вопросы, связанные с разрешающей способностью печатного проводника и целостностью сигнала. Как правило, такие платы соответствую 3 классу точности (минимальная ширина проводника 0,25 мм). Кроме того, можно реализовать конструкцию двусторонней печатной платы по 5 классу точности (минимальная ширина проводника 0,1 мм).

В зависимости от материала основания печатной платы различают многослойные печатные платы на жестком, гибко-жестком или гибком основании. Данная конструкция печатных плат наиболее распространенная. Такая конструкции позволяет решить вопросы не только с целостностью сигнала, за счет применения сигнально-потенциальных звеньев (слои земли и питания, как правило внутренние), но и позволяют увеличить плотность монтажа, за счет увеличения количества слоев, применения глухих и слепых отверстий, функционально сложных электронных компонентов, таких как *BGA* (электронные компоненты, выводы которых расположены под корпусом в виде матрицы шариковых выводов). Кроме того, многослойная структура

позволяет применять оригинальные решения по увеличению плотности монтажа: применение технологии встраивания как активных (бескорпусных), так и пассивных компонентов внутри печатной платы. В настоящее время существуют технологии, способные реализовать многослойную структуру по 6 или даже по 7 классу точности – минимальная ширина проводника 0,050 мм.

Опытным путём было установлено, что все компоненты *IP*-видеокамеры не могут поместиться на односторонней печатной плате с последующей расстановкой проводниковых дорожек. Таким образом, оптимальным решением было выбрать двусторонний тип конструкции печатной платы.

Что касается класса точности, то существует семь классов точности исполнения сложных печатных плат (по ГОСТ Р 53429-2009), они характеризуются наименьшим номинальным значением размеров следующих параметров:

– ширина проводника;

– расстояние между проводниками;

– гарантийный поясок контактной площадки.

Класс точности, также определяет и позиционный допуск на расположение осей отверстий, центров контактных площадок и расположение печатного проводника.

Для проектируемой видеокамеры был выбран класс точности 5, соответственно, ширина проводниковых дорожек должна быть 0.1 мм, минимальное расстояние между проводниками также должно составлять не менее 0.1 мм, Гарантийный поясок контактной площадки – 0,025 мм. Пятый класс точности печатных плат характеризуется высокой точностью и широко применим от производства бытовых приборов до вычислительной техники.

В соответствии с ГОСТ 10317-79, основной шаг координатной сети должен составлять 2.5 мм. При использовании шага координатной сетки менее основного, следует применять шаг, равный 1.25 мм или 0.625 мм. В данном случае при относительно малых размерах как платы, так и компонентов, а также с учетом выбранного класса точности, был выбран шаг 0.625 мм.

### **6.1.2** Выбор и обоснование метода изготовления электронного модуля

Условно все методы изготовления печатных плат можно объединить в три группы:

– аддитивные;

– субтрактивные;

– комбинированные.

Субтрактивный метод производства предусматривают удаление конкретных участков проводящей фольги путем травления. Чаще всего химического. Они применяются, как правило, когда изготавливаются односторонние диэлектрические основания, для которых характерна

избирательная защита рисунка проводников. Может быть применён для создания внутренней прослойки многослойных изделий.

Аддитивный метод предполагает использование нефольгированных диэлектрических оснований, на которые тем или другим способом, избирательно наносят токопроводящий рисунок. Разновидности метода определяются способами металлизации и избирательностью металлизации.

Комбинированные методы объединяют в себе все приемы изготовления печатных плат, необходимые для изготовления печатных проводников и металлизированных отверстий. В зависимости от последовательности операций формирования печатных проводников и металлизированных отверстий различают комбинированный позитивный метод (используются фотошаблоны – позитивы) и комбинированный негативный (используются фотошаблоны – негативы).

Комбинированный позитивный метод позволяет воспроизводить более тонкие проводники за счет меньшей толщины вытравливаемого металла. Толщина используемых в этом методе фоторезистов определяется лишь тем, что толщина рельефа должна быть больше толщины наращиваемой в этом рельефе металлизации (проводников). Данный метод был выбран, так как имеет высокую точность, хорошую надежность изоляции и прочность сцепления металлических элементов с диэлектрическим основанием.

### **6.1.3** Расчёт конструктивно–технологических параметров электронного модуля (определение габаритных размеров, выбор толщины печатной платы, определение элементов проводящего рисунка)

Габаритные размеры устройства определяются габаритами печатной платы и установленной на неё модулями. Проектирование выполнялось согласно заданию, соответственно, длина и ширина устройства – 40мм, высота – 17мм.

Толщина печатной платы, предложенная САПР, осталась 0.4 мм. Материалом диэлектрической основы является стеклотекстолит *FR*4 – наиболее распространённый материал для производства печатных плат. Печатная плата имеет в своём составе 173 дорожки, а также 76 металлизированных отверстия. Это обусловлено использованием двустороннего типа конструкции, а также небольшими размерами платы.

## 6.2 Выбор и обоснование материалов конструкции и защитных покрытий, маркировки деталей и сборочных единиц

Как сказано ранее, в качестве основы печатной платы используется стеклотекстолит *FR*4. Стеклотекстолит типа *FR*4 – это диэлектрик на основе нескольких слоев стеклоткани, пропитанных эпоксидной смолой и имеющий

степень горючести равную нулю. Хорошие диэлектрические свойства, стабильность характеристик и размеров, высокая устойчивость к воздействию неблагоприятных климатических условий.

Защитные или паяльные маски – это специализированные полимерные материалы, наносимые на поверхность печатной платы. Основным назначением паяльной маски является защита токопроводящих поверхностей от случайного замыкания и окисления.

Кроме этого, паяльная маска имеет эстетическое значение, ведь она определяет внешний вид печатной платы [40].

Поверхность всех печатных плат покрыта медными дорожками. Если платы не защищены, медь окисляется, и теряет свои свойства. Для предотвращения окисления меди существуют различные защитные финишные покрытия, а именно:

– Облуживание с выравниванием воздушным ножом (*HASL*, горячее лужение);

– Химический никель / иммерсионное золото (*ENIG*);

– Иммерсионное серебро (*ImAg*);

– Иммерсионное олово (*ImSn*);

– Органическая защитная пассивация (*OSP*).

Покрытие *HASL* – это преобладающее в промышленности финишное покрытие контактных площадок. Процесс заключается в погружении печатной платы и её выдерживание в течение некоторого времени в расплаве оловянно-свинцового или бессвинцового припоя. После выдерживания платы в припое его излишки удаляются «воздушным ножом» – потоком горячего воздуха, продуваемого через отверстия и по всей поверхности платы. Данный тип покрытия выл выбран более предпочтительным в качестве защитного слоя платы.

Маркировку печатных плат, как правило, наносят методом сеткографии специальной краской (с термическим или УФ отверждением, белого, желтого или черного цвета). Разрешение (минимальная толщина линии) такого метода невелика до 0,15мм. Основная маркировка обязательна, содержит в себе обозначение ПП, дату изготовления и обозначение слоя.

# 7 ПРИМЕНЕНИЕ СРЕДСТВ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ПРИ РАЗРАБОТКЕ ПРИБОРА

Как уже было отмечено, в качестве основного САПР для разработки принципиальной электрической схемы использовался *Altium Designer*. Данная программа также позволяет проектировать печатные платы на основе принципиальных схем.

С помощью редактора *PCB* в *AD* был произведён импорт и синхронизация элементов с принципиальной схемой, установлены правила проектирования в соответствии с пятым классом точности. Далее была произведена расстановка элементов, выполнена автоматическая трассировка. К сожалению, зачастую автоматическая расстановка маршрутов срабатывает не идеально, поэтому дальнейшие изменения дорожек проводились вручную. После всех, печатная плата выглядит так (рисунок 26):

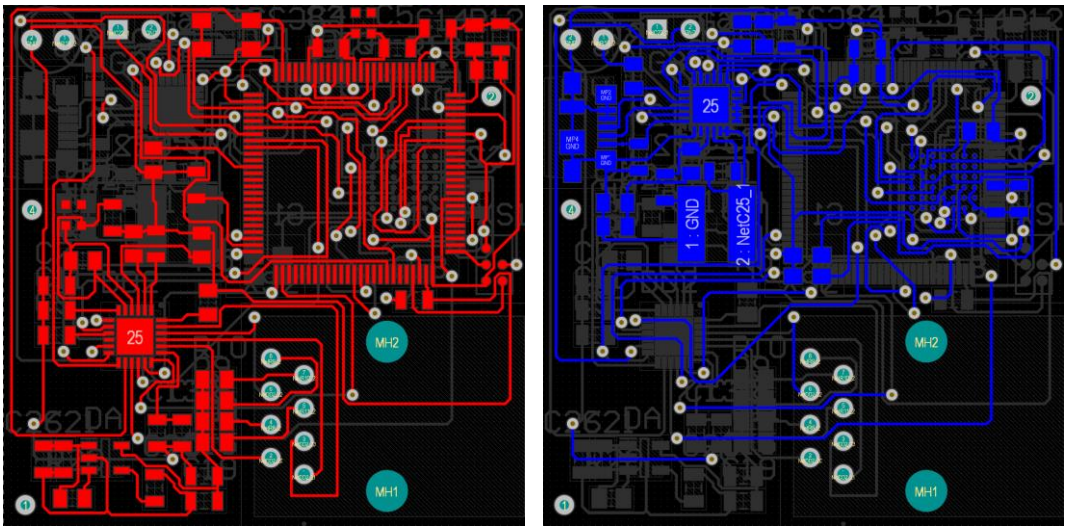



Рисунок 26 – Стороны печатной платы

Далее с помощью сторонней утилиты *Board Assistant* был сформирован чертёж печатной платы, а с помощью встроенного в *AD Draftsman* был сделан сборочный чертёж.

# ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта была спроектирована мобильная *IP*-видеокамера на базе микроконтроллерного ядра *ARM Cortex-M4*.

Видеонаблюдение с использованием *IP*-видеокамер на сегодняшний день актуально как никогда. Системы видеонаблюдения выступают в качестве средства сдерживания преступности, помогают полиции в последующем расследовании. Системы видеонаблюдения защищают посетителей объекта как прямо, так и косвенно, каждого посетителя, который входит в здание, чтобы вести учёт подозрительной активности.

Чтобы создать проект такой видеокамеры была разработана структурная схема, принципиальная схема. На основе этого была разработана печатная плата. Проектирование велось с помощью *Altium Designer*, а также сторонних утилит для данной САПР.

Результатом работы стала печатная плата, которая, согласно заданию, имеет габариты не более 40 на 40 на 20 мм. Таким образом, устройство получилось миниатюрным. Однако, чтобы уместить все компоненты и проводящие дорожки, потребовалось использование двусторонней конструкции печатной платы, а также пятого класса точности, что существенно повысило стоимость производства таких плат.

Что касается программной части работы, то был изучен алгоритм сжатия видео *H.*264, который сейчас повсеместно используется при обработке видео. Были описаны методы, которые использует данный кодек, чтобы снизить размер видео без существенной потери качества. Видеокодеки занимают важное место в контексте локальных сетей, так как необработанные видео могут занять всю пропускную способность сети, а также в принципе занимают много места.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ


[1] Понятие изображения [Электронный ресурс]. – Режим доступа: https://whatis.techtarget.com/definition/image. – Дата доступа: 13.02.2022.

[2] Представление изображений [Электронный ресурс]. – Режим доступа: https://www.teachwithict.com/binary-representation-of-images.html. – Дата доступа: 13.02.2022.

[3] Цветовые модели [Электронный ресурс]. – Режим доступа: https://www.pantone.com/articles/color–fundamentals/color–models–explained. – Дата доступа: 13.02.2022.

[4] Разновидности *RGB* [Электронный ресурс]. – Режим доступа: http://photoschool01.ru/8–or–16–bit. – Дата доступа: 13.02.2022.

[5] Протоколы потокового видео [Электронный ресурс]. – Режим доступа: https://www.cdnetworks.com/media–delivery–blog/video–streaming–protocols. – Дата доступа: 13.02.2022.

[6] Принцип работы видеокамеры [Электронный ресурс]. – Режим доступа: https://eltechbook.ru/kamery_videonabljudenie.html. – Дата доступа: 20.02.2022.

[7] Стандарт *Ethernet* [Электронный ресурс]. – Режим доступа: http://www.highteck.net/EN/Ethernet/Ethernet.html. – Дата доступа: 20.02.2022.

[8] Стек *TCP/IP* [Электронный ресурс]. – Режим доступа: https://isaaccomputerscience.org/concepts/net_internet_tcp_ip_stack?examBoard=all&stage=all. – Дата доступа: 20.02.2022.

[9] Основы *TCP/IP* [Электронный ресурс]. – Режим доступа: https://selectel.ru/blog/tcp-ip-for-beginners. – Дата доступа: 20.02.2022.

[10] Что такое *SoC?* [Электронный ресурс]. – Режим доступа: https://anysilicon.com/what-is-a-system-on-chip-soc. – Дата доступа: 20.02.2022.

[11] *SoC* в видеокамерах [Электронный ресурс]. – Режим доступа: https://www.unifore.net/ip-video-surveillance/ip-camera-dvr-nvrs-soc-whats-it-what-it-does.html. – Дата доступа: 26.02.2022.

[12] Архитектуры *SoC* [Электронный ресурс]. – Режим доступа: https://www.mathworks.com/discovery/soc-architecture.html. – Дата доступа: 26.02.2022.

[13] структура *ARM Cortex-M4* [Электронный ресурс]. – Режим доступа: https://microcontrollerslab.com/arm-cortex-m4-architecture. – Дата доступа: 26.02.2022.

[14] Устройство *GPIO* [Электронный ресурс]. – Режим доступа: https://www.ics.com/blog/introduction-gpio-programming. – Дата доступа: 27.02.2022.

[15] Регистры ядра *ARM Cortex-M4* [Электронный ресурс]. – Режим доступа: https://www.linkedin.com/pulse/all-you-need-know-gpio-akib-islam. – Дата доступа: 27.02.2022.

[16] Как устроен *I2C?* [Электронный ресурс]. – Режим доступа: https://blog.actorsfit.com/a?ID=01750-fa7b01f9-36e1-4b59-b44f-0c32b947d709. – Дата доступа: 27.02.2022.

[17] Регистровая модель *I2C* [Электронный ресурс]. – Режим доступа: https://controllerstech.com/stm32-i2c-configuration-using-registers. – Дата доступа: 03.03.2022.

[18] Про *DCMI* [Электронный ресурс]. – Режим доступа: https://www.st.com/resource/en/application_note/an5020-digital-camera-interface-dcmi-on-stm32-mcus-stmicroelectronics. – Дата доступа: 05.03.2022.

[19] Регистровая модель *DCMI* [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/186980/. – Дата доступа: 05.03.2022.

[20] Принцип работы *DMA* [Электронный ресурс]. – Режим доступа: https://www.minitool.com/lib/direct-memory-access.html. – Дата доступа: 05.03.2022.

[21] Обработка прерываний *DMA* [Электронный ресурс]. – Режим доступа: https://www.eventhelix.com/fault-handling/dma-interrupt-handling. – Дата доступа: 05.03.2022.

[22] Прерывания *DMA* [Электронный ресурс]. – Режим доступа: https://www.brainkart.com/article/DMA-and-Interrupts_8636. – Дата доступа: 05.03.2022.

[22] Модуль камеры *OV*9655 [Электронный ресурс]. – Режим доступа: https://www.arducam.com/downloads/modules/OV9655/ov9655_full. – Дата доступа: 12.03.2022.

[23] интерфейсы *MII* и *RMII* [Электронный ресурс]. – Режим доступа: https://resources.pcb.cadence.com/blog/2019-mii-and-rmii-routing-guidelines-for-ethernet. – Дата доступа: 12.03.2022.

[24] *LAN*8720 [Электронный ресурс]. – Режим доступа: http://ww1.microchip.com/downloads/en/devicedoc/00002165b. – Дата доступа: 12.03.2022.

[25] Введение в *LWIP* [Электронный ресурс]. – Режим доступа: https://cxemotexnika.org/2018/09/vvedenie-v-lightweight-iplwip-stek-protokolov-tcp-ip/. – Дата доступа: 12.03.2022.

[26] Современные стандарты сжатия [Электронный ресурс]. – Режим доступа: https://www.novicam.ru/index.php?route=blog/blog&blog_id=273. – Дата доступа: 13.03.2022.

[27] Что такое *PoE?* [Электронный ресурс]. – Режим доступа: https://www.versatek.com/what-is-power-over-ethernet. – Дата доступа: 13.03.2022.

[28] *PoE* в видеонаблюдении [Электронный ресурс]. – Режим доступа: https://sec-group.ru/blog/ip-kamera-poe. – Дата доступа: 16.03.2022.
[29] Как работает *Li-ion* батарея [Электронный ресурс]. – Режим доступа: https://sec-group.ru/blog/ip-kamera-poe. – Дата доступа: 19.03.2022.
[30] Способы заряда аккумуляторов [Электронный ресурс]. – Режим доступа: https://kit-e.ru/battery/sposoby-zaryada. – Дата доступа: 19.03.2022.
[31] *LTC*4058 [Электронный ресурс]. – Режим доступа: https://www.rlocman.ru/i/File/2017/10/02/405842fs. – Дата доступа: 26.03.2022.
[32] *BQ*2495 [Электронный ресурс]. – Режим доступа: https://www.ti.com/product/BQ24295. – Дата доступа: 27.03.2022.
[33] Альтернативы *AD* [Электронный ресурс]. – Режим доступа: https://www.g2.com/products/altium-designer-2020-05-12/competitors/alternatives. – Дата доступа: 27.03.2022.
[34] *JS API* [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Client-side_web_APIs/Introduction. – Дата доступа: 16.04.2022.
[35] Интерфейс *WebRTC* [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API. – Дата доступа: 16.04.2022.
[36] Интерфейс *WebRTC* [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API. – Дата доступа: 16.04.2022.
[37] Частотное пространство [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/316580. – Дата доступа: 23.04.2022.
[38] Преобразование Фурье [Электронный ресурс]. – Режим доступа: https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm. – Дата доступа: 23.04.2022.
[39] Компенсация движения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/edison/blog/480430. – Дата доступа: 23.04.2022.
[40] Виды паяльных масок [Электронный ресурс]. – Режим доступа: https://belplata.by/produktsiya/materialy-dlya-pechatnykh-plat/zashchitnye-maski. – Дата доступа: 01.05.2022.

**ПРИЛОЖЕНИЕ А**
**(обязательное)**
**Перечень элементов**

**ПРИЛОЖЕНИЕ Б**
**(обязательное)**
**Спецификация**

**ПРИЛОЖЕНИЕ В**
**(обязательное)**
**Визуализированная трёхмерная модель**

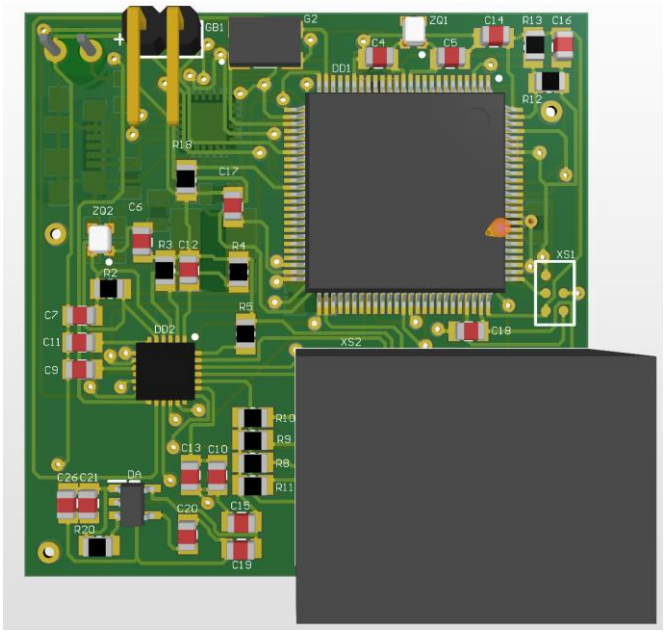



Рисунок В.1 – Верхняя сторона печатной платы

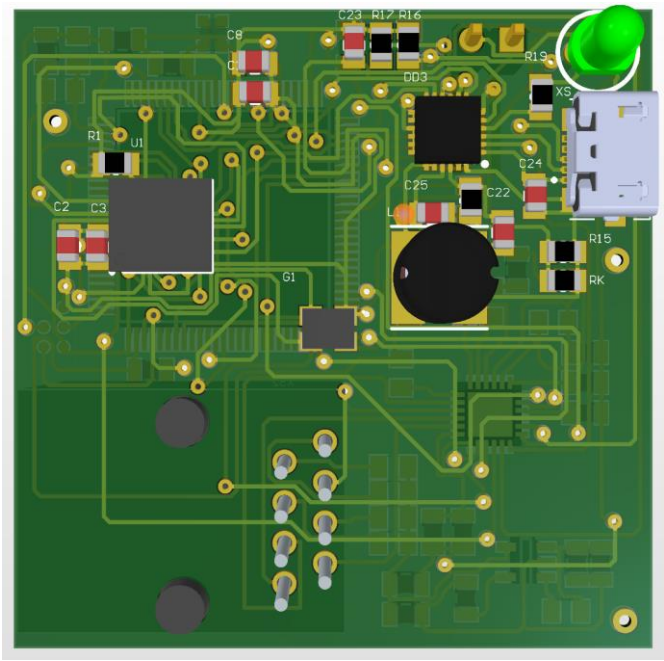



Рисунок В.2 – Нижняя сторона печатной платы

# ПРИЛОЖЕНИЕ Г
## (обязательное)
## Листинг программы

```
import './App.css';
import {useEffect, useRef} from "react";
import React from 'react'
import CanvasJSReact from "./canvasjs-3.6.1/canvasjs.react";
const CanvasJS = CanvasJSReact.CanvasJS;
const CanvasJSChart = CanvasJSReact.CanvasJSChart;
let dps = []
let xVal = dps.length + 1;
let statIter = 0
let yVal = 100000;
let windowWidth = window.innerWidth
if(windowWidth >= 1200) windowWidth = Math.round(windowWidth/2)

const chartOptions = {
    title :{
        text: "Статистика данных видеопотока"
    },
    width: windowWidth * 0.8,
    height: windowWidth * 0.5,
    data: [{
        type: "line",
        dataPoints : dps
    }]
}

function App() {
    let chart = useRef(null)

    useEffect(() => {
        let camera_button = document.querySelector("#start-camera");
        let mediaStreamStats = document.querySelector("#mediaStreamStats");
        let codecAvailable = document.querySelector("#codecAvailable");
        let blobLog = document.querySelector("#blobLog");

        let video = document.querySelector("#video");
        //video.setAttribute("width",`${windowWidth * 0.8}`)
        let videoH264 = document.querySelector("#videoH264");
        let start_button = document.querySelector("#start-record");
        let stop_button = document.querySelector("#stop-record");
        let download_link = document.querySelector("#download-video");

        let camera_stream = null;
        let media_recorder = null;
        let blobs_recorded = [];

        const updateChart = (size) => {
            dps.push({x: xVal,y: size});
            xVal++;
            if (dps.length >  10 ) {
                dps.shift();
            }
            chart.current.render()
            updateStat()
        }
        const updateStat = () => {
```

```
            blobLog.textContent = `
            Current data:\n
            ${statIter}: block size: ${blobs_recorded[statIter].size}, block
type:                ${blobs_recorded[statIter].type}                ,device:
${camera_stream.getVideoTracks()[0].label}
            `
            statIter++
        }

        camera_button.addEventListener('click', async function () {
            //console.log(navigator.mediaDevices.getSupportedConstraints())
            camera_stream = await navigator.mediaDevices.getUserMedia(
                {
                    video: {
                        width: windowWidth * 0.8,
                        facingMode: "environment"
                    },
                    audio: false,
                });
            video.srcObject = camera_stream;
            console.log(camera_stream)
            mediaStreamStats.textContent = `
            Using: ${camera_stream.getVideoTracks()[0].label}
            stream id: ${camera_stream.id}, isActive:${camera_stream.active}`
        });

        start_button.addEventListener('click', function () {
            // set MIME type of recording as video/webm
            if(MediaRecorder.isTypeSupported('video/webm;codecs=H264')) {
                media_recorder = new MediaRecorder(camera_stream,
                    {
                        mimeType: 'video/webm;codecs=H264'
                    });
                codecAvailable.textContent  =  `${media_recorder.mimeType}  is
supported`
            } else {
                media_recorder = new MediaRecorder(camera_stream,
                    {
                        mimeType: 'video/webm;codecs=vp8'
                    });
                codecAvailable.textContent  =  `H264  is  not  supported,  using
${media_recorder.mimeType} instead`
            }

console.log(MediaRecorder.isTypeSupported('video/webm;codecs=H264'))


            // event : new recorded video blob available
            media_recorder.addEventListener('dataavailable', async (e) => {
                blobs_recorded.push(e.data);
                console.log(e.data)
                updateChart(e.data.size)
            })

            // event : recording stopped & all blobs sent
            media_recorder.addEventListener('stop', function () {
                // create local object URL from the recorded video blobs
                download_link.href          =          URL.createObjectURL(new
Blob(blobs_recorded, {type: 'video/webm'}));
            });
```

```
            // start recording with each recorded blob having 1 second video
            media_recorder.start(1000);
        });

        stop_button.addEventListener('click', function () {
            media_recorder.stop();
        });
    })

    return (
        <div className="App d-flex">
            <header>
                <meta   name="viewport"   content="width=device-width,initial-
scale=1.0"/>
                <link
                    rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css
"
                    integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
                    crossOrigin="anonymous"
                />
            </header>
            <div className="d-flex flex-column w-100">
                <div className="d-flex flex-row justify-content-center">
                    <button  id="start-camera"  className="btn  btn-primary  m-
1">Start Camera</button>
                    <button id="start-record" className="btn btn-secondary m-
1">Start Record</button>
                    <button  id="stop-record"  className="btn  btn-secondary  m-
1">Stop Record</button>
                    <a id="download-video" download="test.webm" className="btn
btn-secondary m-1">Download Video</a>
                </div>
                <div className="d-flex flex-column flex-wrap p-2">
                    <div id='mediaStreamStats' className="m-2"/>
                    <video id="video" autoPlay className="flex-item m-2"/>
                    <CanvasJSChart options = {chartOptions}
                                   className="m-2"
                                   ref={chart}
                                   onRef={(ref) => {
                                       chart.current = ref
                                   }}
                    />
                    <div className="m-2 p-2">
                        <div id='codecAvailable'/>
                        <div id='blobLog'/>
                    </div>
                </div>
            </div>
        </div>
    );
}

export default App;
```

**ПРИЛОЖЕНИЕ Д**
**(обязательное)**
**Ведомость документов**