МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. И. КАНТА»
ИНСТИТУТ ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Рекомендована к защите:
методический руководитель
направления подготовки
к.т.н., доцент ИФМНиИТ

_____ М.П. Савченко

"____"_____ 2021 г.

Допущена к защите:
первый заместитель директора
ИФМНиИТ
к. ф.-м. н., доцент

_____А.А. Шпилевой

"____"_____ 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема: «Разработка аппаратно-программной части устройства для незрячих»

Направление подготовки: 11.03.02 «Инфокоммуникационные технологии и системы связи»

Профиль подготовки:
«Многоканальные телекоммуникационные системы»
Квалификация (степень): **бакалавр**

ВКР защищена на оценку:

_____

Выполнил: студент 4 курса

_____ Ю.В. Войтешонок

Руководитель: к.ф.-м.н., доцент ИФМНиИТ
_____ Д.В. Шитц

Рецензент: к.ф.-м.н
_____ И.В. Алексеенко

Рецензент: к.ф.-м.н
_____ А.В. Радиевский

Калининград, 2021

СОДЕРЖАНИЕ

Введение

Необходимость разрабатываемого устройства для незрячих возникла потребностью получения и передачи информации об окружении пользователю без использования органов зрения.

Эта идея не является новой, на рынке присутствуют решения [1] [2], позволяющие в той или иной мере давать пользователю возможность ориентации в пространстве без применения органов зрения. Однако эти решения полагаются на использование внешних служб (GPS, постоянное подключение к сети сотовой связи) либо на заранее построенные карты. Что делает качественную работу устройства зависимым от внешних факторов, времени (даты) создания карты или вовсе невозможной в условиях устаревания карт, отсутствия связи или изменения местности. В некоторых случаях областью действия устройства является открытая местность. Наряду с этим разработаны аппараты для незрячих [3], где 2D изображение преобразуется в звуковой сигнал, и по тональности звука человек в какой-то степени может распознавать окружающее его пространство. Однако устройства такого рода имеют ряд недостатков:

- необходимость долго обучать человека распознавать и привыкать к звуковому зрению;
- в ушах постоянно генерируется звук, который может заглушить другие важные информативные звуки (звонок телефона, звуковой сигнал светофора, сирена скорой помощи, человеческая речь и т.д.);
- большая нагрузка на слуховой орган;
- отсутствие информации в звуковом сигнале об удалённости предметов.

Цель данной работы направлена на устранение этих недочетов. Разрабатываемое устройство должно получать информацию окружающего мира без зависимостей от внешних служб и в реальном времени. Данные, получаемые пользователем, должны основываться на реальном расположении физических объектов, находящихся непосредственно в поле зрения устройства. Информация

передается пользователю посредством тактильной матрицы. Возможность эффекта сенсорного зрения была показана в 1969 г. [4], где на поверхность спины человека тактильно передавались изображения, и человек кожей спины мог различать эти изображения.

Задачей данной работы является разработка системы снятия и вычисления стереоизображения для дальнейшего построения на его основе карты глубины. Для решения задачи снятия и вычисления стереоизображения необходимо провести анализ существующих технологий и алгоритмов получения карты глубины. Определить характеристики и требования технологий и алгоритмов получения карты глубины и выбрать подходящие для поставленной задачи. Проанализировать возможные варианты построения устройства, передающего информацию пользователю.

Для решения данных задач снятия и вычисления стереоизображения и определения характеристик и требований были изучены работы по получению стереоизображений, изучены программные реализации решений инструментов по обработке изображений.

Для решения задачи анализа вариантов построения устройства, передающего информацию пользователю, были изучены возможные варианты конструкции тактильной матрицы, передающей информацию пользователю и изучены работы, касающиеся аппаратной реализации матрицы. В ходе работы были рассмотрены варианты управления матрицей через сдвиговые регистры по протоколу SPI, многоканальные ШИМ-контроллеры, управляющиеся по I^2C.

1. Методы получения стереоизображения

1.1 Методы, основанные на измерении времени распространения сигнала

Для осуществления системы снятия и вычисления стереоизображения для дальнейшего построения карты глубины были задействованы камеры. Они оказались наиболее подходящими с точки зрения доступности и удобства использования. Среди других вариантов построения системы были рассмотрены метод ультразвукового определения расстояния.

1.1.1 Ультразвуковые датчики расстояния

Ультразвуковые датчики расстояния предназначены для бесконтактного измерения расстояния. Принцип их работы основан на технике измерения времени распространения сигнала и достаточно хорошо изучен [5] [6] [7] [8] [9]. Такие датчики достаточно дешевы и работают на расстояниях до нескольких метров, однако возникают проблемы как с их точностью, так и с их поведением в шумных условиях на открытом воздухе.

1.1.1.1 Принцип работы

Расстояние от объекта до датчика рассчитывается как:

$$D = k \cdot T_f \cdot V_s, \quad (1)$$

где T_f - время прохождения ультразвукового импульса, т.е. время, за которое импульс преодолевает расстояние D;

k - константа, зависит от геометрии датчика;

V_s - скорость звука в воздухе.

Как показано в [10] ультразвуковой импульс может быть сгенерирован с помощью пьезоэлектрического преобразователя, а отраженное от объекта отражение принимается другим пьезоэлектрическим преобразователем (рис. 1). Два преобразователя установлены близко друг к другу и составляют измерительную головку.

Поскольку измеренные величины T_f и V_s могут считаться некоррелированными, стандартная неопределенность измеренного расстояния $u(D)$ может быть получена [11] [9] из (1) как:

$$u(D) = \sqrt{\left(k \cdot T_f\right)^2 \cdot u^2(V_s) + (k \cdot T_s)^2 \cdot u^2\left(V_f\right)}, \quad (2)$$

где $u(V_s)$ и $u\left(V_f\right)$ - стандартные неопределенности скорости звука и времени пролета.

Скорость звука в воздухе зависит от температуры θ и, в меньшей степени, от влажности воздуха h

$$V_s = f(\theta, h)$$

Таким образом, из (2):

$$u(D) = \sqrt{\left(k \cdot T_f\right)^2 \cdot \left[\left(\frac{\partial f}{\partial \theta}\right)^2 \cdot u^2(\theta) + \left(\frac{\partial f}{\partial h}\right)^2 \cdot u^2(h)\right] + (k \cdot V_s)^2 \cdot u^2\left(T_f\right)}$$

Если влажность считается случайной величиной, равномерно распределенной в диапазоне от 10% до 90%, ее влияние на скорость звука составляет около 0,15% при 20°C. Это приводит к стандартной погрешности около 0,3 мм для диапазона расстояний 0,3 м, поэтому датчик влажности обычно не требуется.

Скорость звука в воздухе зависит от температуры согласно приближенному уравнению:

$$V_s \approx 20.005 \cdot \sqrt{T},$$

где T - абсолютная температура, измеряемая в кельвинах.

Следовательно, изменения скорости звука в диапазоне 330–360 м/с следует ожидать при изменении температуры в диапазоне 0–40°C. Такой эффект необходимо учитывать при определении расстояния, поэтому требуется датчик температуры.

Для генерации импульса можно использовать ультразвуковые сигналы с частотами в диапазоне от 30 кГц до 5 МГц. Более высокие частоты могут быть предпочтительнее, поскольку они подразумевают более низкие длины волн и, следовательно, потенциально лучшее разрешение, но затухание звука в воздухе резко возрастает с увеличением частоты. Кроме того, более высокие частоты требуют как дорогостоящих преобразователей, так и быстрых электронных устройств, что не позволяет получить дешевую конструкцию. Более низкие частоты имеют то преимущество, что они связаны с низким уровнем рассеяния и могут быть получены с помощью недорогих преобразователей, но длина волны в воздухе составляет несколько миллиметров, что требует особого внимания, чтобы получить погрешности измерения ниже длины волны.

Несколько подходов были исследованы для получения субволновых погрешностей [5] [6] [7] [8] [9] с довольно хорошими результатами, но предлагаемые решения обычно требуют использования некоторой формы цифровой обработки полученных данных, что обычно увеличивает стоимость датчика.

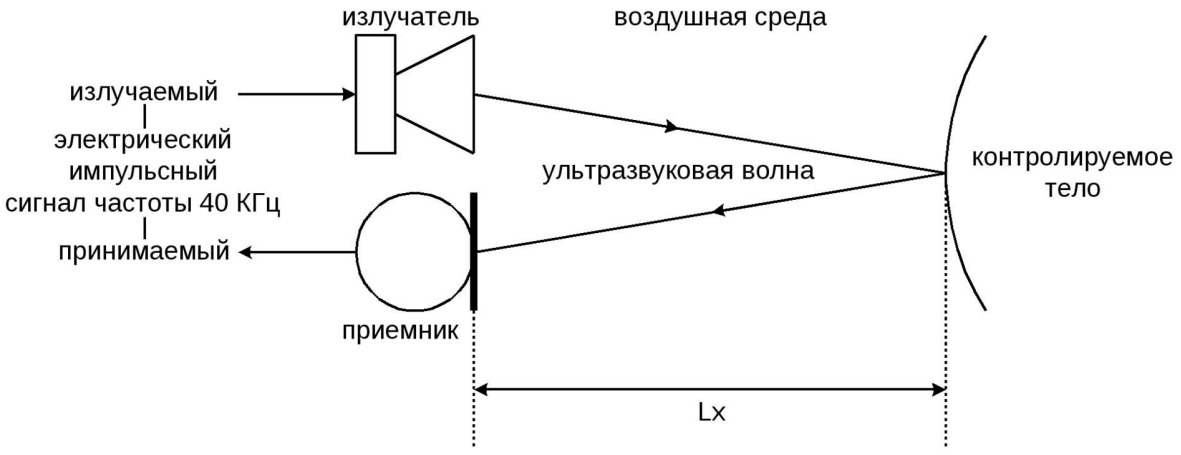

Рис.1. Пример схемы ультразвуковых датчиков расстояния, состоящего из передатчика и приемника

Для разрабатываемого устройства этот тип датчиков имеет недостаточную разрешающую способность: они способны с удовлетворительной погрешностью измерять дальность до объекта, но не способны передавать информацию о самом объекте (форма, рельеф).

1.2 Использование изображений для измерения расстояния

Наиболее подходящей технологией получения стереоизображения для разрабатываемого устройства оказался метод с видеокамерами. Данный метод дает детальную и гладкую карту глубины. При этом рассматривались варианты с использованием одной и двух камер.

1.2.1 SGBM

Для разрабатываемого прототипа необходимо точное и эффективное построение карты глубины в реальном времени. Наиболее сложными в построении и использовании алгоритмов построения карты глубины являются окклюзии, границы объектов и мелкие структуры, которые могут казаться размытыми. Вычисления также затруднены из-за повторяющихся текстур, которые типичны для структурированных сред. Дополнительные практические проблемы возникают из-за различий при снятии изображений и освещении. Большинство алгоритмов для вычисления стереоизображений выполняют четыре шага при работе [12] : вычисление стоимости сопоставления, агрегация стоимости, вычисление (оптимизация) диспаратности (disparity, несоответствие) и обработка диспаратности. Вычисление стоимости сопоставления очень часто основывается на абсолютной, квадратичной или сэмплированной разнице [13] яркости или цветов. Поскольку эта стоимость чувствительна к радиометрическим различиям, также используются функции стоимости, основанные на градиентах изображения. Агрегирование стоимости связывает стоимость в пределах определенного района. Часто стоимость просто суммируются по окну фиксированного размера при постоянной диспаратности. Некоторые методы

дополнительно придают вес каждому пикселю в окне в соответствии с цветовым сходством и близостью к центральному пикселю. Другой подход - выбор окрестности в соответствии с сегментами постоянной яркости или цвета. Вычисление диспаратности выполняется для локальных алгоритмов путем выбора диспаратности с наименьшей стоимостью сопоставления, то есть по принципу победитель получает все. Глобальные алгоритмы обычно пропускают этап агрегирования стоимости и определяют глобальную функцию энергии (energy function), которая включает терм данных и терм сглаживания. Первый суммирует стоимость пиксельного сопоставления, а второй поддерживает плавность диспаратности. В некоторых методах используются дополнительные термы для включения в стоимость окклюзий, альтернативной обработки видимости, обеспечение левого/правого или симметричного соответствия между изображениями или задают вес для терма гладкости в соответствии с информацией о сегментации. Стратегии поиска минимума глобальной энергетической функции различаются. Подходы динамического программирования выполняют оптимизацию в 1D для каждой строки развертки отдельно, что обычно приводит к эффектам полос. Этого можно избежать с помощью методов динамического программирования на основе дерева. Многослойные подходы выполняют сегментацию изображения и плоскости модели в пространстве диспаратности, которые итеративно оптимизируются. Обработка диспаратности часто выполняется для удаления пиков, проверки согласованности, интерполяции промежутков или повышения точности путем субпиксельной интерполяции.

Алгоритмы оценки диспаратности делятся на две большие категории: локальные методы и глобальные методы. Локальные методы оценивают один пиксель за раз, рассматривая только соседние пиксели. Глобальные методы рассматривают информацию, которая доступна во всем изображении. Локальные

методы плохо обнаруживают внезапные изменения глубины и окклюзии, поэтому обычно предпочтительны глобальные методы. Полуглобальное (Semi-global) сопоставление использует информацию от соседних пикселей в нескольких направлениях для вычисления диспаратности пикселя. Но анализ в нескольких направлениях приводит к большим объемам вычислений. Вместо использования всего карты диспаратности пикселя может быть вычислена путем рассмотрения небольшого блока пикселей для упрощения вычислений. Алгоритм SGBM (Semi-Global Block Matching полуглобальное сопоставление блоков) использует [14] блочное сопоставление стоимости, которое сглаживается информацией из нескольких направлений. Стоимость сопоставления вычисляется иерархически с помощью взаимной информации (Mutual Information). Агрегация стоимости выполняется как аппроксимация глобальной энергетической функции путем поэтапной оптимизации со всех сторон изображения. Вычисление диспаратности выполняется по принципу "победитель получает все" и поддерживает такие обработчики диспаратности, как проверка согласованности и интерполяция субпикселей. Многовариантное сопоставление обрабатывается путем объединения диспаратности. Дальнейшие обработка диспаратности включают фильтрацию пиков, выбор согласованной диспаратности по яркости и интерполяцию промежутков.

Глубину в любой заданной точке можно вычислить, если известна диспаратность [15] в этой точке. Диспаратность измеряет смещение точки между двумя изображениями. Чем выше диспаратность, тем ближе объект.

Используя блочный подход, SGBM оценивает приблизительную диспаратность пикселя в левом изображении от того же пикселя в правом изображении.

Уровни диспаратности [16] — это параметр, используемый для определения пространства поиска для сопоставления. Как показано рис. 2,

алгоритм ищет каждый пиксель в левом изображении среди D пикселей в правом изображении. Сгенерированные значения D представляют собой уровни диспаратности D для пикселя в левом изображении. Первые столбцы D левого изображения не используются, поскольку соответствующие пиксели в правом изображении недоступны для сравнения. На рис. 2 w представляет ширину изображения, а h - высоту изображения. Для заданного разрешения изображения увеличение уровня диспаратности уменьшает минимальное расстояние для определения глубины. Увеличение уровня диспаратности также увеличивает вычислительную нагрузку алгоритма. При заданном уровне диспаратности увеличение разрешения изображения увеличивает минимальное расстояние для определения глубины. Увеличение разрешения изображения также увеличивает точность оценки глубины. Количество уровней диспаратности пропорционально разрешению входного изображения для обнаружения объектов на той же глубине.

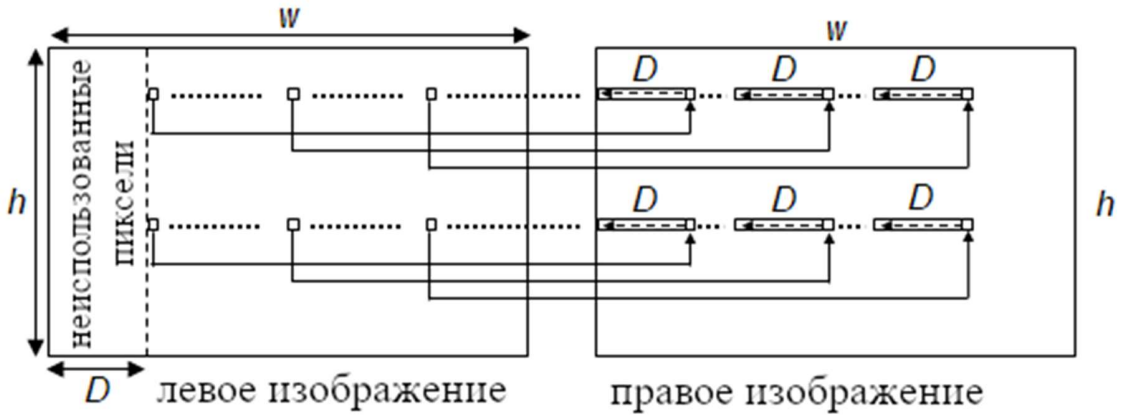

Рис. 2. Работа SGBM для двух входных изображений (w - ширина изображения, h - высоту изображения)

В алгоритме SGBM для оптимизации функции стоимости входное изображение рассматривается с нескольких направлений. В общем, точность полученной диспаратности улучшается с увеличением количества направлений.

Алгоритм описывается отдельными этапами обработки. Некоторые из них не являются обязательными, в зависимости от приложения. Далее будут рассмотрены наиболее характерные этапы.

Алгоритм SGBM принимает в качестве входных данных пару выпрямленных (калиброванных) изображений. Перед снятием стереоизображений с камер необходимо произвести калибровку, чтобы все эпиполярные линии были параллельны горизонтальной оси и соответствовали вертикальным координатам каждого соответствующего пикселя.

Как правило, программная реализация алгоритма SGBM состоит из трех основных модулей [15]: расчет стоимости сопоставления, расчет стоимости согласно направлениям (агрегирование стоимости) и постобработка.

В качестве примера реализации расчета стоимости сопоставления используется преобразование переписи (census transform) [17]. Этот модуль можно разделить на две части: центрально-симметричное преобразование переписи (CSCT) левого и правого изображений и вычисление расстояния Хэмминга. Сначала модель вычисляет CSCT на каждом изображении с помощью скользящего окна. Для выбранного пикселя рассматривается окно размером 9 на 7 пикселей вокруг него. CSCT для центрального пикселя в этом окне оценивается путем сравнения значения каждого пикселя с его соответствующим центрально-симметричным пикселем в окне. Если значение пикселя больше, чем его соответствующий центрально-симметричный пиксель, результатом будет 1, иначе результат будет 0. На рис. 3 показан пример окна 9 на 7. Номер центрального пикселя равен 31. 0-й пиксель сравнивается с 62-м пикселем (синий цвет), 1-й пиксель сравнивается с 61-м пикселем (красный) и так далее. Каждый результат выводится в виде одного бита, а результат для всего окна представляет собой 31-битное число. Это 31-битное число является выходным сигналом CSCT для каждого пикселя в обоих изображениях.

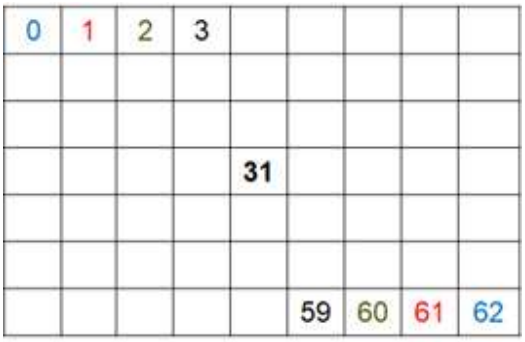

Рис. 3. Пример окна 9 на 7

В модуле расстояния Хэмминга выходы CSCT левого и правого изображений подвергаются пиксельной операции XOR, а установленные биты подсчитываются для генерирования стоимости сопоставления для каждого уровня диспаратности. Для генерации уровней диспаратности D используются блоки вычисления D попиксельного расстояния Хэмминга. Стоимость сопоставления для уровней диспаратности D в заданной позиции пикселя, p, в левом изображении вычисляется путем вычисления расстояния Хэмминга с позициями (от p до D + p) пикселей в правом изображении. Стоимость сопоставления C (p, d) вычисляется в каждой позиции пикселя p для каждого уровня диспаратности d. Стоимость сопоставления не вычисляется для позиций пикселей, соответствующих первым столбцам D левого изображения.

Попиксельный расчет стоимости, как правило, неоднозначен, и неправильные сопоставления могут легко иметь меньшую стоимость, чем правильные, из-за шумов и т.д. Поэтому добавляется дополнительное ограничение, которое поддерживает гладкость, штрафуя изменения соседних диспаратностей. Попиксельная стоимость и ограничения для гладкости выражаются путем определения функции энергии $E(D)$, которая зависит от карты диспаратности D:

$$E(D) = \sum_{p} (C(p, D_p) + \sum_{q \epsilon N_p} P_1 T[|D_p - D_q| = 1] + \sum_{q \epsilon N_p} P_2 T[|D_p - D_q| > 1]) \quad (3)$$

Первое выражение представляет собой сумму всех затрат на сопоставление пикселей для диспаратности D. Второе выражение добавляет постоянный штраф P_1 для всех пикселей q в окрестности N_p точки p, для которых диспаратность немного изменяется (т. е. 1 пиксель). Третье выражение добавляет больший постоянный штраф P_2 за все большие изменения диспаратности. Использование меньшего штрафа за небольшие изменения позволяет адаптироваться к наклонным или изогнутым поверхностям. Постоянный штраф за все большие изменения (т.е. независимо от их размера) сохраняет разрывы на изображении. Разрывы часто видны при изменении яркости. Это используется путем адаптации P_2 к градиенту яркости, то есть $P_2 = \frac{P_2'}{|I_{bp} - I_{bq}|}$ для соседних пикселей p и q в базовом изображении I_b. Однако всегда необходимо следить за тем, чтобы $P_2 \geq P_1$.

Задача стереосопоставления теперь может быть сформулирована как нахождение изображения несоответствия D, которое минимизирует функцию энергии $E(D)$. К сожалению, такая глобальная минимизация, т.е. в 2D, является NP-полной задачей для многих энергий, сохраняющих непрерывность. С другой стороны, минимизация по отдельным строкам изображения (в 1D), может быть эффективно выполнена за полиномиальное время с помощью динамического программирования [13]. Однако решения, использующие техники динамического программирования легко страдают от появления полос из-за сложности соотнесения одномерных оптимизаций отдельных строк изображения

друг с другом в двухмерном изображении. Проблема заключается в том, что очень сильные ограничения в одном направлении - вдоль строк изображения, сочетаются с отсутствием или гораздо более слабыми ограничениями в другом направлении - вдоль столбцов изображения.

Эта проблема решается благодаря агрегированию затрат сопоставления в одномерном измерении по всем направлениям одинаково. Агрегированные (сглаженные) затраты S (p, d) для пикселя p и диспаратности d вычисляются путем суммирования затрат всех направлений 1D с минимальной стоимостью, которые заканчиваются в пикселе p при диспаратности d, как показано на рис. 4. Эти пути, проходящие через пространство несоответствия, проецируется в виде прямых линий на базовое изображение, но в виде непрямых линий в соответствующее изображение сопоставления в соответствии с изменениями диспаратности вдоль путей. Примечательно, что требуется только стоимость пути, а не сам путь.

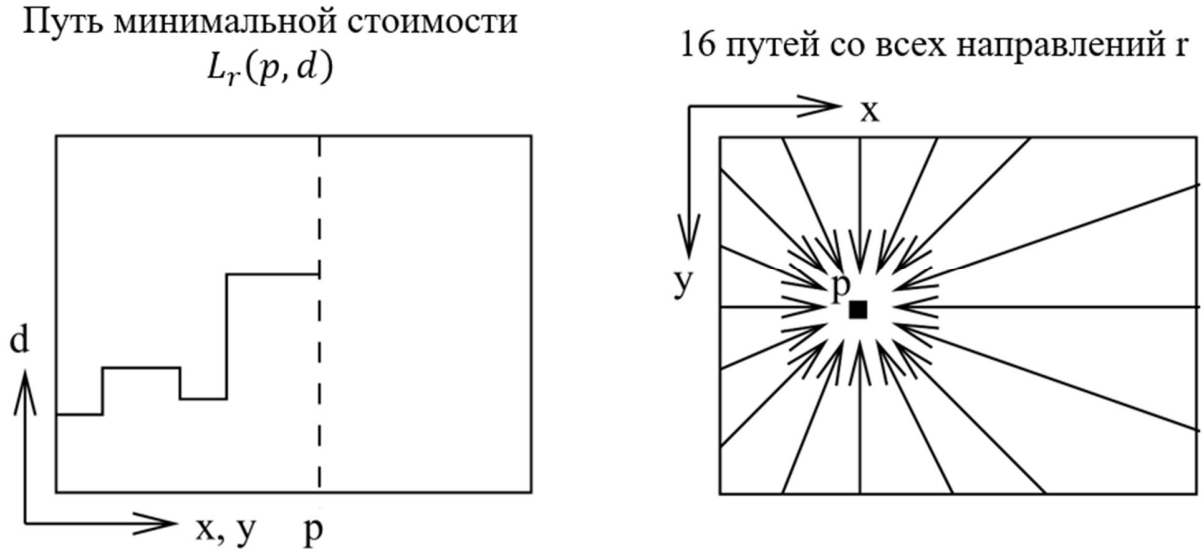

Рис. 4. Агрегация стоимости

Стоимость пути, пройденного в направлении r пикселя p при несоответствии d, определяется рекурсивно как

$$L'_r(p,d) = C(p,d) + \min\left(L'_r(p-r,d), L'_r(p-r,d-1) + P_1, L'_r(p-r,d+1) + P_1, \min_i L'_r(p-r,i) + P_2\right), \quad (4)$$

где $C(p,d)$ - стоимость сопоставления пикселя p и диспаратность d;

$L'_r(p-r,d-1)$ - прошлая стоимость пикселя в направлении r при диспаратности d – 1;

$L'_r(p-r,d+1)$ - прошлая стоимость пикселя в направлении r при диспаратности d + 1;

$\min_i L'_r(p-r,i)$ – минимальная стоимость пикселя в направлении r для предыдущего вычисления;

P1, P2 – штраф за отсутствие непрерывности.

Стоимость пиксельного сопоставления C зависит от конкретной реализации. Остальная часть уравнения добавляет наименьшую стоимость предыдущего пикселя пути p - r, включая соответствующий штраф за разрывы. Данная формула реализует поведение (3) на произвольном одномерном пути.

Значения L' постоянно увеличиваются по пути, что может привести к очень большим значениям. Однако (4) можно изменить, вычтя минимальную стоимость пути предыдущего пикселя из всего уравнения:

$$L_r(p,d) = C(p,d) + \min\left(L_r(p-r,d), L_r(p-r,d-1) + P_1, L_r(p-r,d+1) + P_1, \min_i L_r(p-r,i) + P_2\right) - \min_k L_r(p-r,k), \quad (5)$$

где $L_r(p,d)$ – текущая стоимость пикселя p и диспаратность d в направлении r;

$C(p,d)$ - стоимость сопоставления пикселя p и диспаратность d;

$L_r(p-r,d-1)$ - прошлая стоимость пикселя в направлении r при диспаратности d – 1;

$L_r(p-r,d+1)$ - прошлая стоимость пикселя в направлении r при диспаратности d + 1;

$\min_i L_r(p-r,i)$ – минимальная стоимость пикселя в направлении r для предыдущего вычисления;

P1, P2 – штраф за отсутствие непрерывности.

Это не изменяет фактический путь через пространство диспаратности, поскольку вычитаемое значение является постоянным для всех диспаратностей пикселя p. Таким образом, положение минимума не меняется. Однако теперь верхний предел может быть задан как $L \leq C_{max} + P_2$.

Lr суммируются по путям во всех направлениях r. Количество путей должно быть не менее 8 и обычно составляет 16 для обеспечения хорошего покрытия 2D-изображения. В последнем случае пути, которые не являются горизонтальными, вертикальными или диагональными, реализуются путем перехода на один шаг по горизонтали или вертикали, за которым следует один шаг по диагонали.

Из-за шума результат сопоставления стоимости неоднозначен, и некоторые неправильные сопоставления могут иметь меньшую стоимость, чем правильные. Следовательно, требуются дополнительные ограничения для повышения плавности за счет штрафов за изменения соседних неравенств. Это ограничение реализуется путем объединения одномерных путей с минимальной стоимостью из нескольких направлений. Он представлен агрегированной стоимостью из r направлений в каждой позиции пикселя, $S(p,d)$:

$$S(p,d) = \sum_r L_r(p,d)$$

Верхний предел для S легко определяется как $S \leq 16\ (C_{max} + P_2)$ для 16 путей.

Распространение в каждом направлении независимо. Результирующие диспаратности на каждом уровне с каждого направления суммируются для каждого пикселя. Общая стоимость — это сумма стоимости, рассчитанная по каждому направлению.

Вычисление (5) требует O(D) шагов на каждый пиксель, поскольку минимальная стоимость предыдущего пикселя, например $\min_k L_r(p - r,k)$, постоянна для всех диспаратностей пикселя и может быть вычислена заранее. Каждый пиксель посещается ровно 16 раз, что дает общую сложность O(WHD). Обычная структура и простые операции, такие как операции добавления и сравнения, позволяют выполнять параллельные вычисления с использованием инструкций ассемблера $SIMD^2$ на основе целых чисел.

Третий модуль алгоритма SGBM — это постобработка. Этот модуль состоит из трех шагов: расчет индекса минимальной стоимости, интерполяция и функция уникальности. Расчет индекса минимальной стоимости находит индекс, соответствующий минимальной стоимости для данного пикселя. Квадратичная интерполяция субпикселей применяется к индексу для вычисления диспаратности на уровне субпикселей. Функция уникальности обеспечивает надежность вычисленного значения минимальной диспаратности. Более высокое значение порога для функции уникальности увеличивает количество диспаратности, помеченных как ненадежные. На последнем этапе отрицательные значения диспаратности отбрасываются и заменяются на -1.

1.2.2 Block-matching

Библиотека OpenCV содержит [18] также алгоритм вычисления карты глубины, основанный на сопоставлении блоков (block-matching), он был описан работе [19].

Как было упомянуто выше, методы сопоставления можно классифицировать как локальные или глобальные. Локальные методы пытаются сопоставить небольшие области одного изображения с другим на основе внутренних особенностей региона. Глобальные методы дополняют локальные методы, учитывая физические ограничения, такие как непрерывность поверхности. Локальные методы могут быть дополнительно классифицированы по тому, сопоставляют ли они дискретные признаки (features) изображений или выполняют соотношение небольшой области. Признаки обычно выбираются так, чтобы они не зависели от освещения и точки обзора, например, углы — это хороший элемент для использования в данных целях, потому что углы сохраняются почти во всех проекциях. Алгоритмы на основе признаков компенсируют изменения точек обзора и различия камер и могут обеспечить быстрое и надежное сопоставление. Но у них есть недостаток, заключающийся в том, что они требуют ресурсозатратного извлечения признаков.

Алгоритм, основанный на соотношении областей, сравнивает небольшие участки среди изображений с помощью корреляции. Поскольку данный метод не нуждаются в вычислении признаков и имеют чрезвычайно регулярную алгоритмическую структуру, его реализация может быть оптимизирована.

Типичный метод корреляции областей имеет пять этапов:

1. Коррекция геометрии. На этом этапе искажения входных изображений исправляются путем преобразования в «стандартную форму».
2. Преобразование изображения. Преобразование каждого пикселя изображения в градациях серого в более подходящую форму, например, нормализует его на основе средней локальной яркости.
3. Соотношение областей. Это этап корреляции, на котором каждая небольшая область сравнивается с другими областями в окне поиска.

4. Выделение экстремумов. Определяется крайнее значение корреляции в каждом пикселе, что дает карту диспаратности: каждое значение пикселя представляет собой диспаратность между левым и правым фрагментами изображения при наилучшем совпадении.
5. Постфильтрация. Один или несколько фильтров убирают шум в результирующей карте диспаратности.

Фундаментальной геометрией для стереокорреляции является эпиполярная кривая, показанная на рис. 5. Рассмотрим точку p на одном изображении. Луч из точки p через точку фокусировки захватывает объекты в сцене на разной глубине. Проекция этого луча на втором изображении - эпиполярная кривая, связанная с p. Геометрическое значение этой кривой состоит в том, что для любого объекта, отображаемого в точке p, соответствующая точка на втором изображении должна лежать на эпиполярной кривой. Таким образом, при стереосопоставлении достаточно искать по этой кривой.

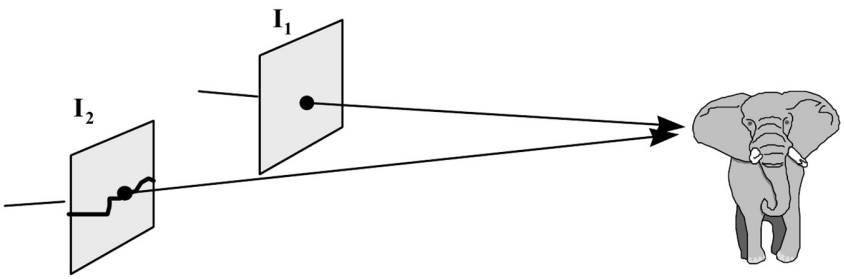

Рис. 5. Эпиполярная кривая — это изображение в I_2 луча, спроецированного из точки I_1.

Если изображения являются идеальными проекциями, снятыми с пинхола, то эпиполярные кривые будут прямыми линиями. Для эффективной обработки лучше всего выровнять эпиполярные линии по правильному шаблону, обычно это линии сканирования камеры. Для этого изображения должны находиться в одной плоскости, с горизонтально выровненными линиями сканирования и

центрами изображений и с одинаковым фокусным расстоянием. Говорят, что стереоизображения с такой геометрией имеют *стандартную форму*. На практике трудно точно достичь этой физической геометрии, и в некоторых случаях необходима вергенция изображений для просмотра близких объектов. Однако, если известны все параметры внутренней и внешней камеры, можно деформировать каждое изображение, чтобы оно отображалось в стандартной форме. На практике полная калибровка стереосистемы трудна и требует много времени, а повторная калибровка, вызванная вибрацией или другими механическими помехами, нецелесообразна. Компромисс заключается в размещении камер приблизительно с правильной геометрией и калибровке внутренних параметров камеры.

Корреляция областей изображения нарушается из-за различий в снимках, освещении и перспективе между изображениями. Методы корреляции областей обычно пытаются компенсировать их путем корреляции не исходных изображений яркости, а некоторого преобразования яркости. Есть три основных типа преобразований:

1. Нормирование яркости. Каждое из значений яркости в коррелированной области нормировано на среднюю яркость в этой области.
2. Лапласиан гауссиана (LoG - Laplacian of Gaussian). Лапласианские измерения направлены на яркости краев на некоторой площади, сглаженной гауссианой. Обычно стандартное отклонение гауссиана составляет 1-2 пикселя.
3. Непараметрический. Эти преобразования - попытка решить проблему выбросов, которые имеют тенденцию подавлять меру корреляции, особенно с использованием разницы квадратов.

Каждое из этих преобразований может применяться с разными мерами корреляции. Например, корреляции LoG включают соответствие пересечений нуля и их знаков, а также две меры величины: квадрат и абсолютную разность (норма L1).

Алгоритм, реализованный для вычисления карты глубины содержит следующие характеристики:

- преобразование LoG, корреляция по норме L1 (абсолютная разность);
- переменный размер поиска диспаратности (16, 24 или 32 пикселя);
- постфильтрация с оператором интереса (interest operator);
- 4x интерполяция.

LoG преобразование и норма L1 дают хорошее качество и могут быть оптимизированы с помощью стандартных наборов инструкций, доступных для DSP и микропроцессоров [19].

1.3 Получение стереоизображения при помощи одной камеры

Оценка глубины с помощью одной камеры включает в себя несколько методов, которые разрабатывались на протяжении многих лет. Одним из наиболее успешных среди этих методов является SfM (Structure-from-Motion, структура по движению) [20]; он использует движение камеры для оценки позиции камеры через различные временные интервалы и, в свою очередь, оценивает глубину с помощью триангуляции из пар последовательных изображений. В качестве альтернативы SfM, для оценки глубины можно использовать другие методы, такие как вариации освещения [21] или фокусировки [22].

1.3.1 FCRN

Методы оценки глубины с помощью одной камеры полагаются на некоторые допущения, связанные с окружающей средой. Для получения карты

глубины из одного изображения в режиме реального времени была рассмотрена возможность применения нейронной сети архитектуры FCRN, основанной на CNN (convolutional neural network, свёрточная нейронная сеть).

Архитектура FCRN основана на ResNet-50 с заменой полносвязных слоев на блоки повышения разрешения (up-sampling, слой повышения дискретности), дающих на выходе примерно половину входного разрешения.

Почти все современные архитектуры CNN содержат сужающуюся часть, которая постепенно снижает разрешение входного изображения посредством серии сверток и слоев объединения (pooling), давая нейронам более высокого уровня большие рецептивные поля, таким образом захватывая более глобальную информацию. В задачах регрессии, в которых желаемым выходом является изображение с высоким разрешением, требуется некоторая форма повышения разрешения, чтобы получить более крупную карту выходных данных. В работах [23] [24] использовались полносвязные слои, создавая полное рецептивное поле. Затем результат преобразуется в выходное разрешение.

В FCN рецептивное поле является важным аспектом архитектурного дизайна, поскольку здесь нет явных полных связей. Для примера можно рассмотреть входное изображение размером в 304 × 228 пикселей [24] и выходную карту, которая будет иметь примерно половину входного разрешения. Для примера, рецептивное поле на последнем сверточном слое AlexNet [25] 151 × 151 пиксель, что позволяет получать только входные изображения с очень низким разрешением, когда истинная глобальная информация должна быть захвачена сетью без полносвязных слоев. Большее рецептивное поле 276 × 276 достигается с помощью VGG16 [26], но устанавливает предел входного разрешения. В работе [23] продемонстрировано существенное улучшение при переходе с AlexNet на VGG, но поскольку обе модели используют полносвязные слои.

В ResNet [27] были представлены обходящие связи (skip layers), которые обходят две или более свертки и суммируются с их выходными данными, включая пакетную нормализацию [28] после каждой свертки (см. рис. 6). Следуя этой архитектуре, можно создавать гораздо более глубокие сети, не сталкиваясь с деградацией или проблемой исчезающих градиентов. Еще одним полезным свойством этих чрезвычайно глубоких архитектур является их большое рецептивное поле; ResNet-50 захватывает входные размеры 483 x 483, что достаточно для полного захвата входного изображения даже в более высоких разрешениях. Учитывая выбранный размер ввода и эту архитектуру, последние сверточные слои приводят к 2048 картам признаков (feature maps) с пространственным разрешением 10 × 8 пикселей при удалении последнего слоя объединения. Если вместо этого был добавлен полносвязный слой того же размера, он содержал бы 3,3 миллиарда параметров суммарным объемом 12,6 ГБ памяти, что сделало бы этот подход сложным в реализации. Это еще больше мотивирует идею о полностью сверточной архитектуре с блоками повышения разрешения, которые содержат меньше весов, но при этом повышают точность прогнозируемых карт глубины.

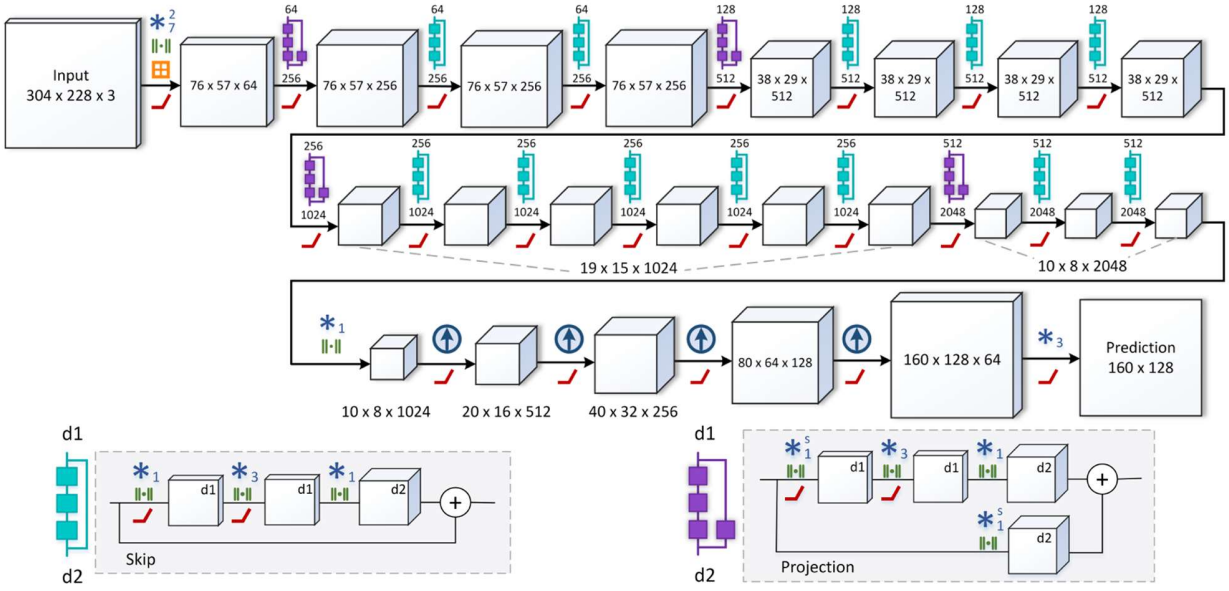

Рис. 6. Архитектура FCRN, где: $*_n^{(s)}$ - свертка n×n со страйдом s; - ReLU; - слой объединения по максимуму (max-pooling) 3 × 3 со страйдом 2; - пакетная нормализация; - up-convolution

Размеры карты признаков соответствуют сети, обученной для входного размера 304 × 228, в случае набора данных NYU Depth v2 [29]. Первая часть сети основана на ResNet-50 и инициализируется предварительно обученными весами. Вторая часть архитектуры управляет сетью для обучения ее масштабированию с помощью последовательности развертывающего слоя (unpooling, расщепление, обратное к объединению) и сверточных слоев. После набора этих блоков повышения разрешения применяется dropout (метод прореживания, метод исключения), за которым следует последний сверточный слой, дающий ответ.

Развертывающие слоя [30] [31] [32] выполняют операцию, обратную к объединению, увеличивая пространственное разрешение карт признаков. В этой архитектуре адаптирован подход, описанный в [30], для реализации развертывающих слоев, чтобы удвоить размер, отображая каждую запись в верхний левый угол ядра 2 × 2. За каждым таким слоем следует свертка 5 × 5 - так что она применяется к более чем одному ненулевому элементу в каждом

месте и применяется активация ReLU. Эмпирически, складываются четыре таких блока с развёртками (up-convolution), то есть 16-кратное масштабирование наименьшей карты признаков, что приводит к наилучшему компромиссу между потреблением памяти и разрешением. Эмпирически было установлено, что при добавлении пятого блока производительность не увеличилась.

Идея состоит в том, чтобы ввести простую свертку 3 × 3 после развертки и добавить проецированное соединение из карты признаков с более низким разрешением к результату, как показано на рис. 7 (c). Из-за разных размеров карта небольшого размера должна быть увеличена с помощью другой развертки в ветви проекции, но поскольку расщепление нужно применить только один раз для обеих ветвей, мы просто применяем свертки 5 × 5 отдельно для двух ветвей. Этот новый блок развертки назван блоком повышения проекции, поскольку он расширяет идею связи проекции [27] для развёртки. Объединение блоков повышения проекции позволяет более эффективно передавать высокоуровневую информацию по сети, постепенно увеличивая размеры карты признаков. Это позволяет построить согласованную, полностью сверточную сеть для прогнозирования карты глубины. На рис. 7 показаны различия между блоком развертки и блоком повышения проекции.

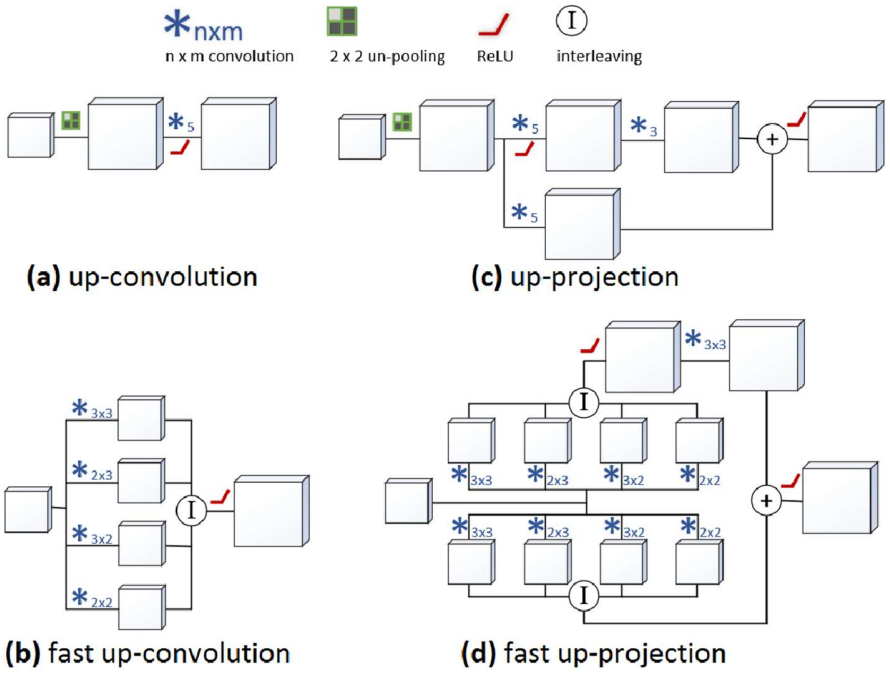

Рис. 7. Варианты построения развёртки:

a) стандартная развёртка (up-convolution); b) более быстрая развёртка;

c) используемый в FCRN блок повышения проекции (up-projection);

d) более быстрый блок повышения проекции.

Быстрая развертка более эффективна, чем обычная развертка. Это приводит к сокращению времени обучения всей сети примерно на 15%. Основная идея заключается в следующем: после расщепления 75%, итоговая кара признаков содержит нули, поэтому следующая свертка 5 × 5 в основном работает с нулями, которых можно избежать в модифицированной развертке. Это можно увидеть на рис. 8. В левом верхнем углу исходная карта признаков не расщепляется (вверху в середине), а затем сворачивается с помощью фильтра 5 × 5. Можно наблюдать, что на не расщепленной карте признаков, в зависимости от расположения (красный, синий, фиолетовый, оранжевый) фильтра 5 × 5, только определенные веса умножаются на потенциально ненулевые значения. Эти веса делятся на четыре неперекрывающиеся группы, обозначенные разными цветами и A, B, C, D на рисунке. Основываясь на группах фильтров, можно объединить

исходный фильтр 5 × 5 в четыре новых фильтра размеров (A) 3 × 3, (B) 3 × 2, (C) 2 × 3 и (D) 2 × 2. Тот же результат, что и при первоначальной операции (расщепление и свертка), теперь может быть получен путем чередования элементов четырех итоговых карт признаков, как на рис. 8.

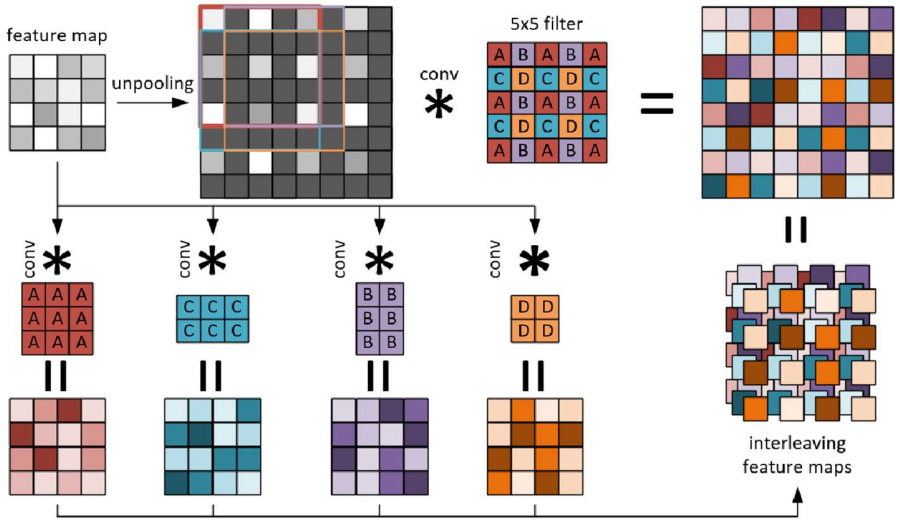

Рис. 8. Быстрая развертка. Верхний ряд: обычные шаги развёртки: при расщеплении размер карты признаков увеличивается вдвое, пустоты заполняются нулями, а свертка 5 × 5 фильтрует эту карту. В зависимости от положения фильтра, только определенные его части (A, B, C, D) умножаются на ненулевые значения. Это мотивирует к свертке исходной карты признаков с 4 различными фильтрами (нижняя часть) и их чередование для получения одного и того же вывода, избегая при этом нулевого умножения. A, B, C, D только отмечают расположение, фактические значения веса будут отличаться.

В качестве функции потерь для задач регрессии обычно используют норму L2 (RMSE, $\|\cdot\|, \|\cdot\|_2$), минимизирующий квадрат евклидовой нормы между предсказанными значениями $\tilde{y}$ и истинными y:

$$\mathcal{L}_2(\tilde{y} - y) = ||\tilde{y} - y||_2^2$$

Это дает хорошие результаты для данных тестовых примерах, но было обнаружено, что использование обратного Хубера (reverse Huber, berHu) [33] $\mathcal{B}$ в качестве функции потерь дает лучшую окончательные результаты, чем L2.

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases}$$

Потери BerHu равны норме $\mathcal{L}_1(x) = |x|$, когда x ∈ [−c, c] и равно L2 вне этого диапазона. Используемая версия является непрерывной и дифференцируемой по первому порядку в точке c, где происходит переход с L1 на L2. На каждом этапе градиентного спуска, когда мы вычисляем $\mathcal{B}(\tilde{y} - y)$, мы устанавливаем

$$c = \frac{1}{5}\max_i(|\tilde{y}_i - y_i|),$$

где i проходит по всем пикселям каждого изображения в текущем пакете, что составляет 20% от максимальной ошибки для каждого пакета. Эмпирически BerHu показывает хороший баланс между двумя нормами в данной проблеме; он придает высокий вес выборкам / пикселям с высокими остатками из-за L2, в отличие, например, от двухвесовой функции Тьюки (Tukey), которая игнорирует выборки с высокими остатками [34]. В то же время L1 учитывает большее влияние меньших градиентов остатков, чем L2. В обоих наборах данных, с которыми были проведены эксперименты, наблюдалось распределение значений глубины с тяжелым хвостом (heavy-tailed distribution), для которого Звальд и Ламберт Лакруа [33] показали, что функция потерь berHu более подходящая.

Были получены количественные и качественные результаты [35], оценка производилась на таких датасетах, как NYU Depth v2 (внутри помещения) и Make3D [36] (вне помещения). В данной работе эксперименты проводились на

GTX TITAN (12 Гб). Весовые уровни части down-sampling архитектуры инициализируются соответствующими моделями (AlexNet, VGG, ResNet), предварительно обученными на данных ILSVRC для классификации изображений. Вновь добавленные слои части up-sampling инициализируются как случайные фильтры, отобранные из нормального распределения с нулевым средним и дисперсией 0,01.

Сеть обучается на RGB изображениях для прогнозирования соответствующих карт глубины. Было использовано увеличение данных, чтобы увеличить количество обучающих выборок. Входные изображения и соответствующее истинное изображение преобразуются с использованием небольших поворотов, масштабирования, цветовых преобразований и переворачиваний с вероятностью 0.5. Далее моделируются небольшие переводы путем случайной обрезки расширенных изображений до выбранного размера входа в сеть.

Был оценен один из крупнейших наборов данных RGB-D для реконструкции сцены в помещении, NYU Depth v2. Набор необработанных данных состоит из 464 сцен, снятых с помощью Microsoft Kinect, с официальным разделением, состоящим из 249 обучающих и 215 тестовых сцен. Однако для обучения методу требуется лишь небольшое подмножество необработанного распределения. Были отобраны кадры с равным интервалом из каждой обучающей последовательности, в результате чего получается примерно 12 тысяч уникальных изображений. После офлайн-дополнений (аугментаций, augmentations) извлеченных кадров набор данных включает примерно 95 тысяч пар изображений RGB-D. Полученный набор данных радикально меньше, чем тот, который требуется для обучения модели в [23] [24], состоящий из 120 тыс. уникальных изображений, а также 800 тыс. сэмплов, извлеченных с помощью patch-wise. Исходные кадры размером 640 × 480 пикселей подвергаются

понижению размерности до 1/2 и обрезаются по центру до 304 × 228 пикселей, как входные данные в сеть. Далее модель обучается с размером пакета 16 примерно за 20 эпох. Начальная скорость обучения - 10^{-2} для всех слоев, которая уменьшается каждые 6-8 эпох, когда наблюдается плато; момент 0,9.

Для количественной оценки методов и сравнения с датасетом были вычислены различные меры ошибок на часто используемом тестовом подмножестве из 654 изображений. Размер прогнозов зависит от конкретной модели. В данной конфигурации, которая состоит из четырех этапов повышения размерности, соответствующие выходные разрешения составляют 128 × 96 для AlexNet, 144 × 112 для VGG и 160 × 128 для моделей на основе ResNet. Затем предсказания подвергаются повышению размерности до исходного размера (640 × 480) с использованием билинейной интерполяции и сравниваются с предоставленной истинными значениями с заполненными значениями глубины для недопустимых пикселей.

В таблице 1 сравнивается различные варианты CNN предлагаемой архитектуры, чтобы изучить влияние каждого компонента. Сначала были оценены влияние глубины архитектуры с помощью сверточных блоков AlexNet, VGG-16 и ResNet-50. Очевидно, что полностью сверточная архитектура (UpConv) в AlexNet уступает типичной сети с полными соединениями (FC). Причиной этого является относительно небольшое поле зрения в AlexNet, которого недостаточно для сбора глобальной информации, необходимой при удалении полносвязных слоев. Вместо этого использование VGG в качестве базовой архитектуры повышает точность оценки глубины. Поскольку вариант с полной связью VGG для многомерной регрессии будет включать большое количество параметров, были проведены тесты только на полностью сверточной (UpConv) модели. Однако модель на основе VGG с полностью связанными

слоями использовалась в [23] (таблица 2) и работала лучше, чем полностью сверточный вариант VGG.

Таблица 1. Сравнение разных подходов с различными вариантами в наборе данных NYU Depth v2. Для ошибок rel, rms, log10 чем ниже, тем лучше, для точности $\delta_i < 1.25^i$ чем больше, тем лучше.

Архитектура	Функция потерь		Параметры	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
AlexNet	FC	$\mathcal{L}_2$	104.4×10^6	0.209	0.845	0.090	0.586	0.869	0.967
		berHu		0.207	0.842	0.091	0.581	0.872	0.969
	UpConv	$\mathcal{L}_2$	6.3×10^6	0.218	0.853	0.094	0.576	0.855	0.957
		berHu		0.215	0.855	0.094	0.574	0.855	0.958
VGG	UpConv	$\mathcal{L}_2$	18.5×10^6	0.194	0.746	0.083	0.626	0.894	0.974
		berHu		0.194	0.790	0.083	0.629	0.889	0.971
ResNet	FC-160x128	berHu	359.1×10^6	0.181	0.784	0.080	0.649	0.894	0.971
	FC-64x48	berHu	73.9×10^6	0.154	0.679	0.066	0.754	0.938	0.984
	DeConv	$\mathcal{L}_2$	28.5×10^6	0.152	0.621	0.065	0.749	0.934	0.985
	UpConv	$\mathcal{L}_2$	43.1×10^6	0.139	0.606	0.061	0.778	0.944	0.985
		berHu		0.132	0.604	0.058	0.789	0.946	0.986
	UpProj	$\mathcal{L}_2$	63.6×10^6	0.138	0.592	0.060	0.785	0.952	0.987
		berHu		**0.127**	**0.573**	**0.055**	**0.811**	**0.953**	**0.988**

Таблица 2. Сравнение различных подходов [37] [38] [39] [40] [41] [42] [24] [43] [23] для датасета NYU Depth v2.

NYU Depth v2	rel	rms	rms(log)	$\log_{10}$	δ_1	δ_2	δ_3
Karsch	0.374	1.12	-	0.134	-	-	-
Ladicky	-	-	-	-	0.542	0.829	0.941
Liu	0.335	1.06	-	0.127	-	-	-
Li	0.232	0.821	-	0.094	0.621	0.886	0.968
Liu	0.230	0.824	-	0.095	0.614	0.883	0.971
Wang	0.220	0.745	0.262	0.094	0.605	0.890	0.970
Eigen	0.215	0.907	0.285	-	0.611	0.887	0.971
Roy and Todorovic	0.187	0.744	-	0.078	-	-	-
Eigen and Fergus	0.158	0.641	0.214	-	0.769	0.950	**0.988**
ResNet-UpProj	**0.127**	**0.573**	**0.195**	**0.055**	**0.811**	**0.953**	**0.988**

Переход на ResNet с полностью подключенным уровнем (ResNet-FC) - без удаления последнего уровня объединения - обеспечивает производительность, аналогичную [23] для вывода с низким разрешением (64 × 48), используя в 10 раз меньше данных; однако увеличение выходного разрешения (160 × 128) приводит к огромному количеству параметров. Это дополнительно мотивирует аргументы в пользу замены полносвязных слоев и необходимости в более эффективных методах повышения размерности при работе с изображениями большой размерности. Рассматриваемый сверточный вариант с использованием простых сверток вверх (ResNet-UpConv) повышает точность, а предлагаемая архитектура (ResNetUpProj), усиленная блоками повышения проекции, дает наилучшие результаты. Наблюдается резкое уменьшение количества параметров при переходе от полносвязных слоев к полностью сверточным сетям. Другой распространенный метод повышения размерности, — это деконволюция (deconvolution) с последовательными ядрами 2 × 2, но повышение проекции заметно превзошло его. Поскольку рассматриваемый метод состоит из четырех

последовательных шагов повышения размерности (2-кратное разрешение на блок), он может сохранить больше структуры на выходе по сравнению с вариантом FC (рис. 9).

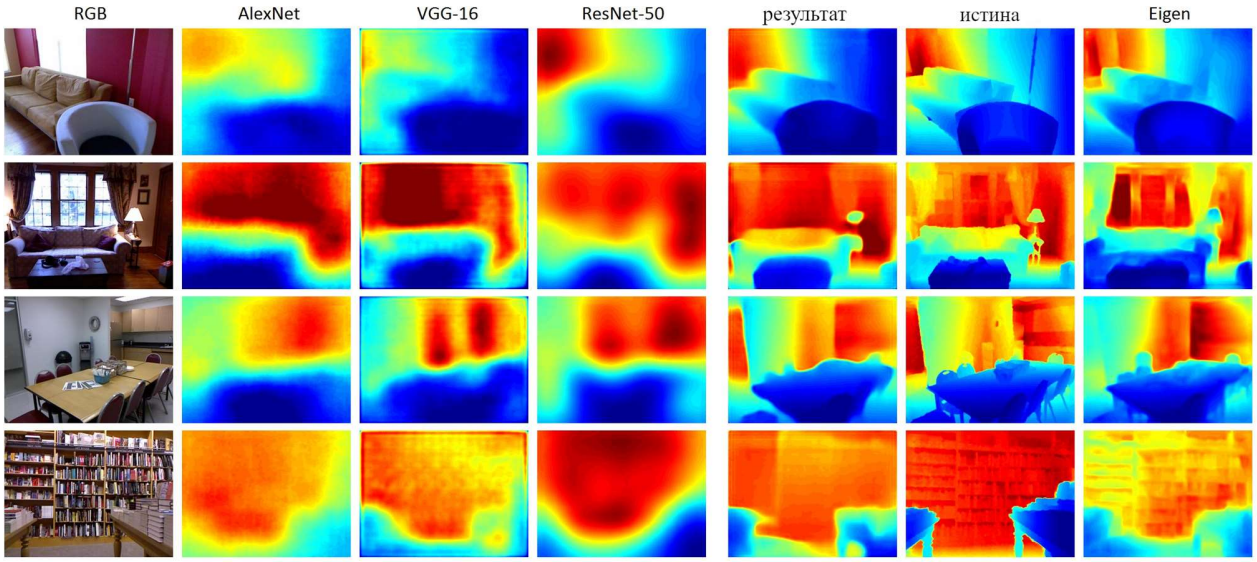

Рис. 9. Результаты на наборе NYU Depth. Результаты, полученные с использованием AlexNet, VGG и полносвязного ResNet по сравнению с рассматриваемой моделью и прогнозами [23]. Все карты масштабируются одинаково для лучшего сравнения.

Во всех тестах потери berHu превосходят L2. Разница в относительной ошибке выше, что может быть объяснено большими градиентами L1 (berHu) над L2 для малых остатков; влияние на относительную ошибку выше, поскольку пиксели на меньших расстояниях более чувствительны к меньшим ошибкам. Этот эффект также хорошо виден как более сильный выигрыш в мере δ_1. Было измерено время одиночного сверточного блока для одного изображения (1,5 мс) и сравнено с повышением проекции (0,14 мс). Одним из преимуществ данной модели является общее время вычислений. Прогнозирование карты глубины одного изображения занимает всего 55 мс с предложенным повышением размерности (78 мс с up-convolution). Это позволяет обрабатывать изображения в реальном времени, например, с веб-камеры. Дальнейшее ускорение может быть

достигнуто при одновременной обработке нескольких изображений. Время вычисления для размера пакета, равного 16, составляет 14 мс на изображение с увеличением проекцией и 28 мс для up-convolution.

В таблице 2 сравниваются результаты, полученные с помощью предлагаемой архитектуры, с результатами, полученными в соответствующих работах. Кроме того, на рис. 9 показывается качественное сравнение точности расчетных карт глубины с использованием предлагаемого подхода (ResNet-UpProj) с точностью различных вариантов (AlexNet, VGG, ResNet-FC64x48), а также с общедоступными прогнозами Eigen [23]. Можно видеть улучшение качества от AlexNet к ResNet, однако полносвязный вариант ResNet, несмотря на его повышенную точность, по-прежнему ограничен грубыми прогнозами. Данная полностью сверточная модель значительно улучшает качество границ и четкость структуры на прогнозируемых картах глубины.

Прогнозы глубины демонстрируют примечательное визуальное качество, даже если они получены с помощью одной модели, обученной от начала до конца, без каких-либо дополнительных шагов постобработки, как, например, вывод CRF из [40] [42]. С другой стороны, [23] уточняет свои прогнозы с помощью многомасштабной архитектуры, которая объединяет изображение RGB и исходное прогнозирование для создания визуально привлекательных результатов. Однако они иногда неверно оценивают глобальный масштаб (вторая и третья строки) или вносят шум в случае сильно текстурированных областей в исходном изображении, даже если фактическая граница глубины отсутствует в истинном изображении (последняя строка). Кроме того, в [5] количество параметров, примерно равняется 218 миллиона, что примерно в 3,5 раза больше, чем в рассматриваемой модели. Вместо этого, данная архитектура CNN разработана с учетом реализуемости - количество параметров не должно бесконтрольно увеличиваться в задачах большой размерности. Это

дополнительно означает сокращение количества требуемых шагов градиента, а также выборок данных, необходимых для обучения. Единая сеть лучше обобщает и успешно решает проблему грубости, с которой сталкивались предыдущие подходы CNN к прогнозированию карты глубины.

2. Способы построения виброматрицы

Для поставленной задачи был выбран метод построения виброматрицы на основе вибромоторов. Этот метод обладает рядом достоинств, таких как доступность и дешевизна используемых компонентов.

В ходе работы изучались варианты управления виброматрицей на основе сдвиговых регистров, а также многоканальные ШИМ котроллеры для LED диодов PCA9685.

2.1 Виброматрица на основе сдвиговых регистров

Для прототипа использовались сдвиговые регистры в качестве задающего элемента интенсивности вибрации отдельного вибромотора. Для данных целей была выбрана микросхема sn74hc595 серии 7400 производства Texas Instruments. Сдвиговые регистры управляются с помощью интерфейса SPI. За счет архитектуры sn74hc595 [44] (рис. 11) и особенностей передачи данных по интерфейсу SPI [45] [46] (рис. 10), получилось добиться простого сопряжения управляющей части и виброматрицы (рис. 12).

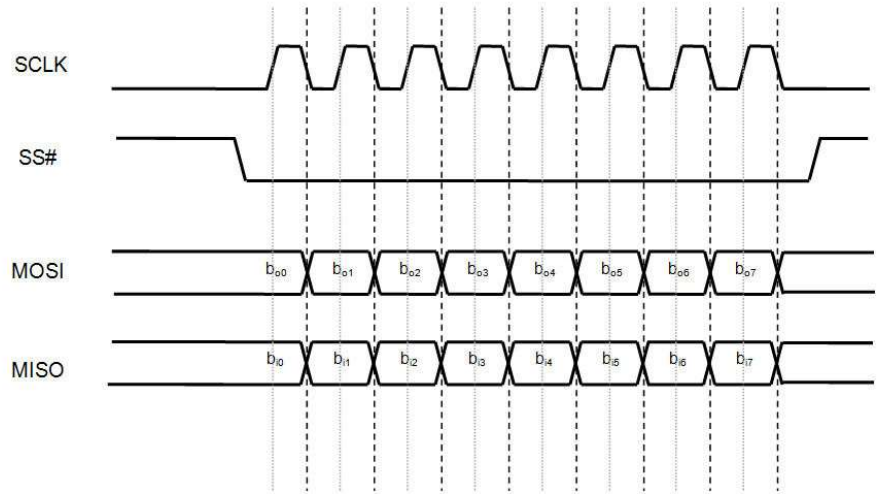

Рис. 10. Передача данных по интерфейсу SPI

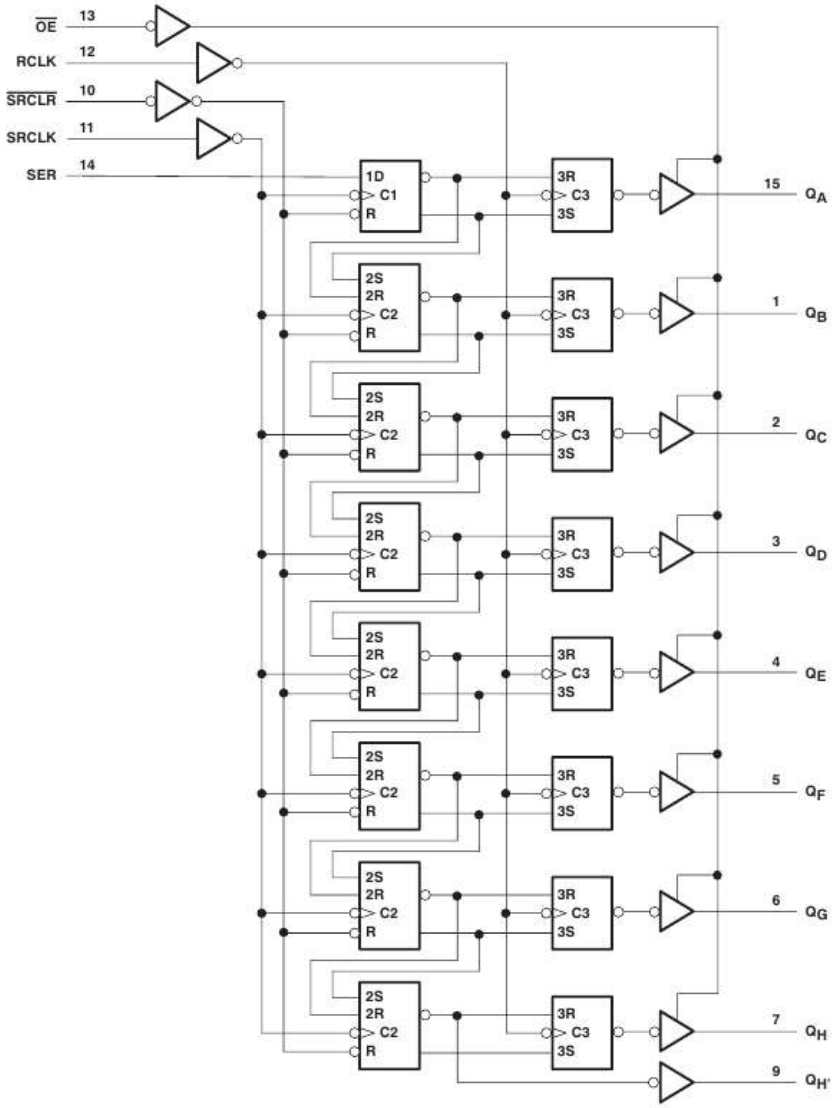

Рис. 11. Логическая схема sn74hc595

Так, сигнал синхронизации поступает на вход SRCLK (shift register clock, тактирование сдвигового регистра) микросхемы sn74hc595, который подается на входы синхронизации каждого разряда. Если посмотреть на форму передачи данных по интерфейсу SPI (рис. 10), то можно заметить, что с сигналом синхронизации (SCLK) одновременно передается импульс данных по линии MOSI (Master Out Slave In, выход ведущего, вход ведомого). Линия передачи MISO (Master In Slave Out вход ведущего, выход ведомого) не используется в данном проекте, так как нет необходимости в считывании данных. Таким

образом, данные с интерфейса SPI последовательно вносятся во входные триггеры через вход SER (serial input, последовательных вход), а затем с сигналом SS (Slave Select, выбор ведомого, также CS Chip Select, выбор микросхемы), который поступает на вход RCLK (storage register clock, тактирование регистра хранения), переносится в группу выходных регистров, которые сохраняют состояние до следующего сигнала SS. Таким образом, при достаточно высоких частотах тактирования сдвиговых регистров, выбор которой зависит от количества вибромоторов, возможно управление моторами посредством ШИМ, где скважность зависит от передаваемой информации (формируется программно), а частота от частоты тактирования сдвиговых регистров и количества моторов.

Для масштабирования количества вибромоторов, возможно последовательное подключение сдвиговых регистров (рис. 13) с последовательным переносом данных следующему регистру через вывод Q_H' (QH_ на схеме), который подключен к последовательному входу следующей микросхемы. В это случае сигнал SS должен передаваться с окончанием одной посылки, когда все данные внесены в триггеры первого ряда.

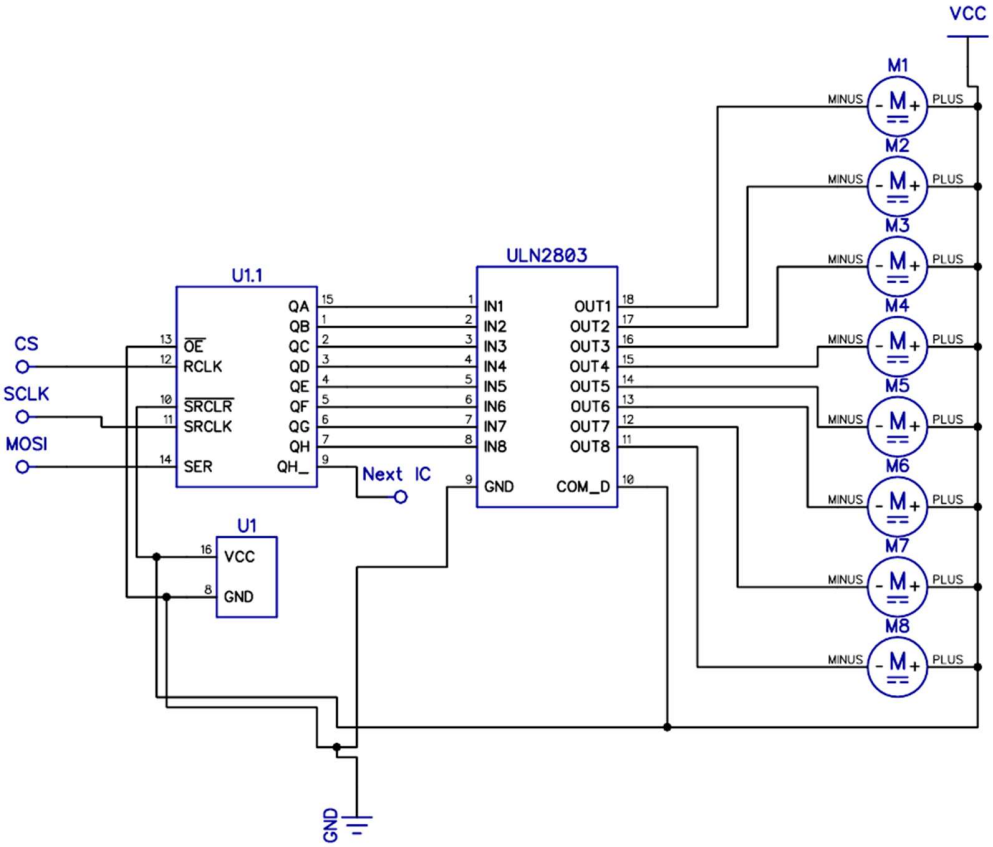

Рис. 12. Схема управления первыми 8 вибромоторами;

CS, SCLK, MOSI – входы для интерфейса SPI, Next IC – выход для последовательного переноса данных следующему регистру

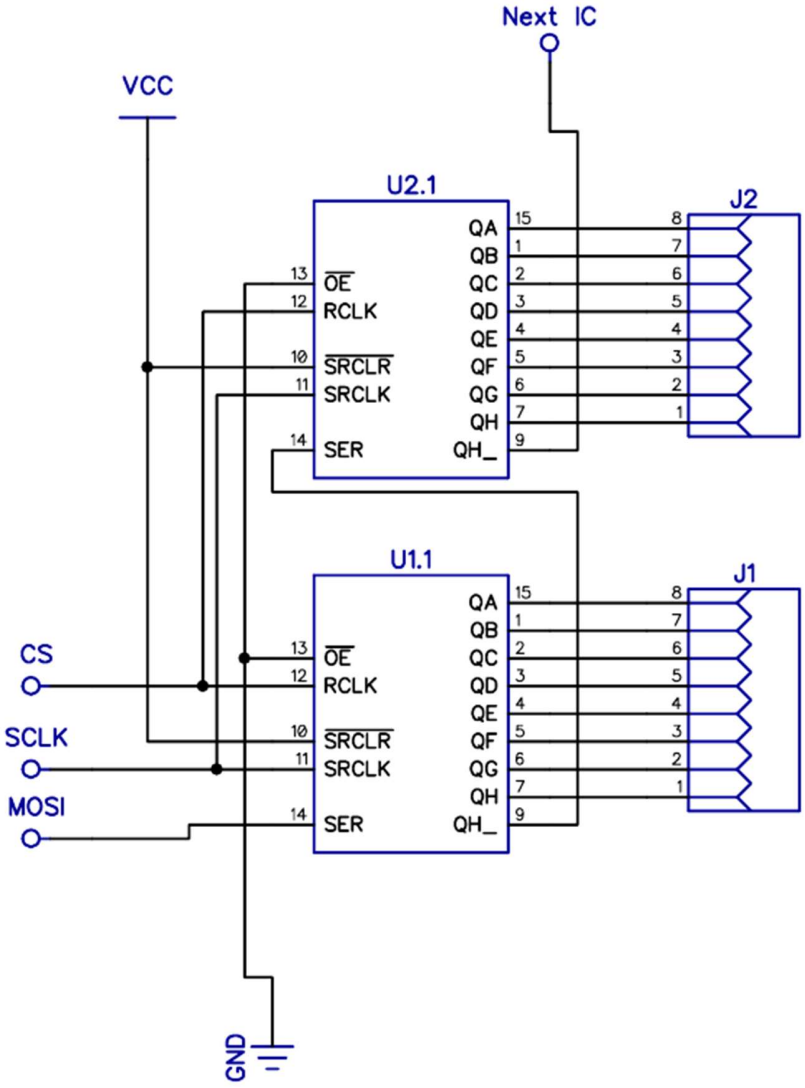

Рис. 13. Схема последовательного включения сдвиговых регистров; CS, SCLK, MOSI – входы для интерфейса SPI Next IC – выход для последовательного переноса данных следующему регистру, J1, J2 – выходы для подключения вибромоторов (через сборки Дарлингтона)

Данная схема показывает основную идею получения 64 (матрица 8X8) независимых ШИМ каналов для управления вибромоторами.

В данном случае главным ограничением дальнейшего увеличения разрешения матрицы является максимальная скорость тактирования сдвиговых

регистров (рис. 14). Увеличение регистров влечет за собой увеличение длительности одного ШИМ-импульса. Таким образом, частота ШИМ:

$$f_{PWM} = \frac{f_{clock}}{Nn_{PWM}},$$

где:N – число используемых регистров;

n_{PWM} – число разрядов ШИМ-сигнала.

		V_{CC}	T_A = 25°C		SN54HC595		SN74HC595		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
f_{clock}	Clock frequency	2 V		6		4.2		5	MHz
		4.5 V		31		21		25	
		6 V		36		25		29	

Рис. 14. Максимальные частоты тактирования sn74hc595 [44]

В прототипе в качестве драйвера матрицы, который получает сигнал по USB от модуля вычисления карты глубины и далее передает сигнал на сдвиговые регистры, использовался микроконтроллер stm32f103c8t6 (ARM 32-bit Cortex-M3). Из схемы тактирования данного микроконтроллера [47] [48] (рис. 15) видно, что максимальная скорость тактирования блока SPI достигается блоком SPI1 на линии APB1 (48 Мгц). Что после делителей (рис. 16, рис. 17) позволяет развить максимальную скорость передачи данных по SPI до 18 Мбит/с. Что соответствует максимальной скорости блока SIP для данной серии микроконтроллеров.

Таким образом, для 64-разрядной матрицы с разрядностью ШИМ 256 f_{PWM} равняется 1099 Гц. Что является допустимым, но ограничивает масштабируемость матрицы (для частот, меньших 100 Гц становится заметны колебания вибромоторов). В дальнейших разработках применяются многоканальные контроллеры ШИМ, которые не обладают зависимостью частоты ШИМ от частоты передачи данных.

Для более эффективного использования в микроконтроллере задействован контроллер DMA (прямой доступ к памяти). Он используется для ускорения процесса перемещения данных между периферийными блоками и основной

памятью. Данные перемещаются контроллером DMA без какого-либо участия процессора.

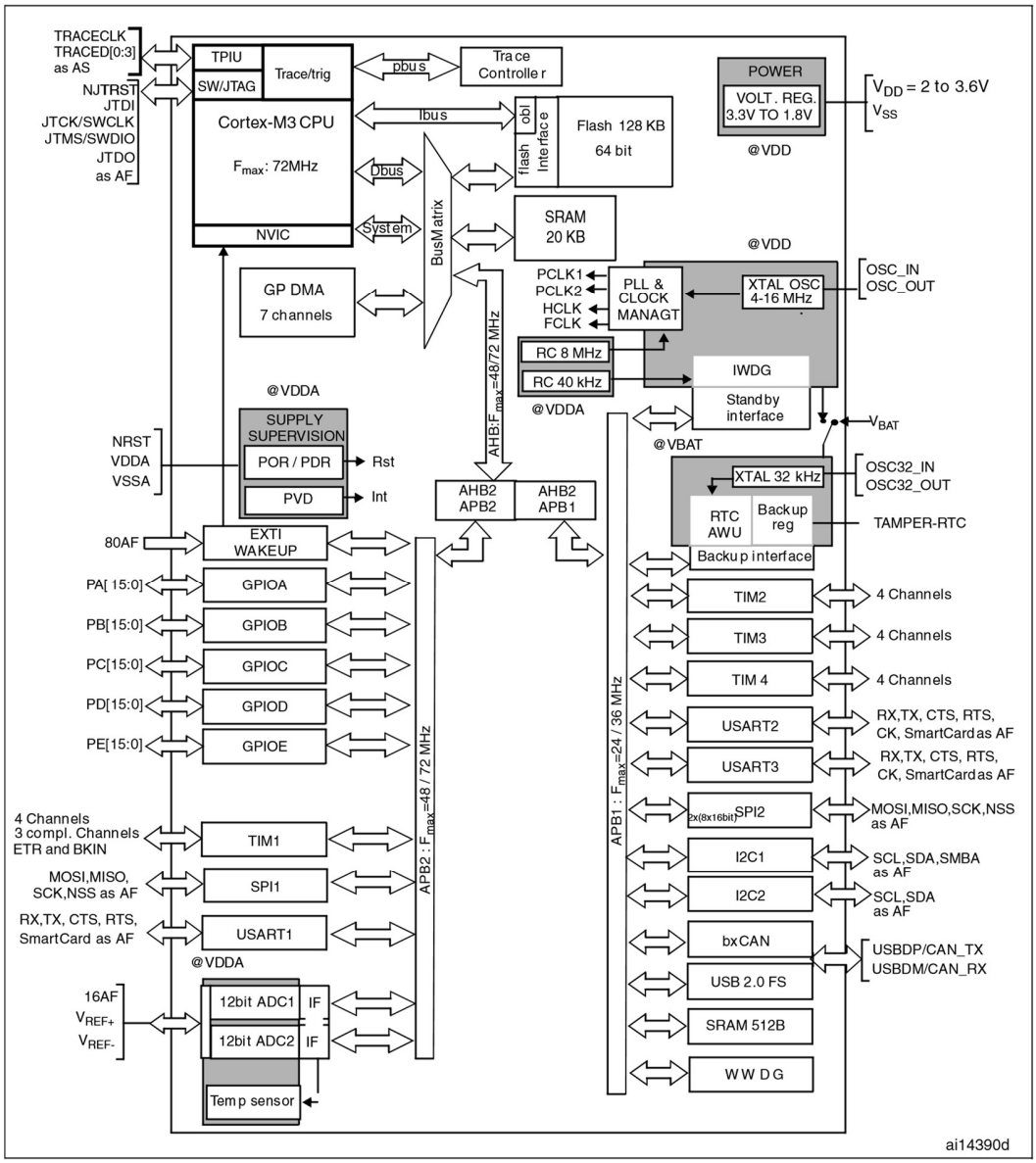

Рис. 15. Блок-схема серии МК STM32F103xx

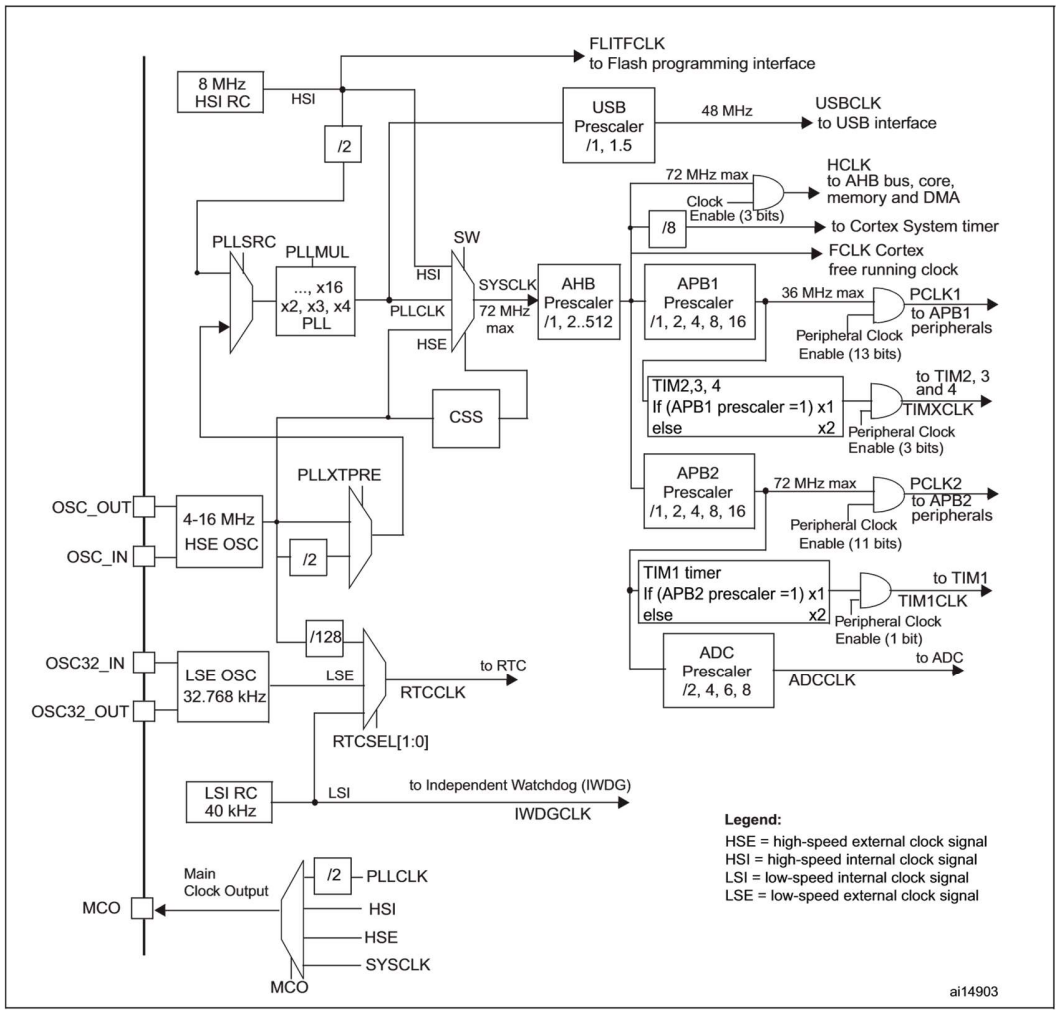

Рис. 16. Дерево тактирования серии МК STM32F103

Передача ведется в полудуплексном режиме – блок SPI работает всегда только на передачу. Тогда из рис. 19 необходимо задействовать 3 канал - SPI1_TX. Для подтверждения передачи и дальнейшей отправки следующего блока данных необходимо следить за регистром DMA_ISR, флаг TCIFx. Согласно рис. 18 необходим 9 бит (3 канал). В DMA_ISR каждый бит отображает:

31:28 – зарезервированные биты, должны оставаться в сброшенном состоянии

27, 23, 19, 15,11, 7, 3 – TEIFx флаг ошибки передачи (TE, transfer error) для канала x. 0 - нет ошибок передачи, 1 – случилась ошибка передачи

26, 22, 18, 14, 10, 6, 2 – HTIFx флаг прохождения половины передачи (HT, half transfer) для канала x. 0 – прохождения половины передачи не произошло, 1 – произошло прохождение половины передачи.

25, 21, 17, 13, 9, 5, 1 – TCIFx флаг завершения передачи (TC, transfer complete) для канала x. 0 – передача не завершена, 1 - передача завершена.

24, 20, 16, 12, 8, 4, 0 – GIFx флаг глобального прерывания, устанавливается при событиях TC, HT, TE для канала x. 0 – не было TC, HT, TE, 1 – произошло TC, HT, TE.

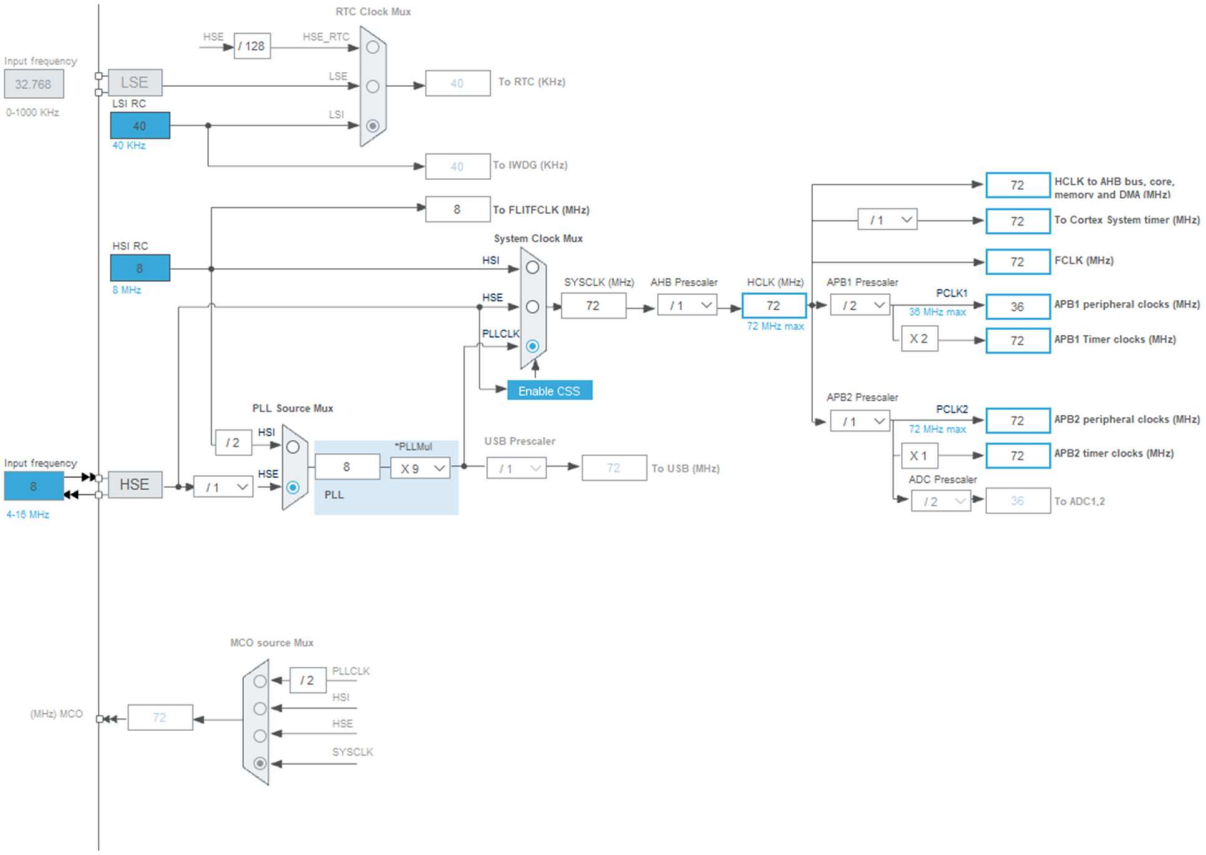

Рис. 17. Значения делителей для получения максимальной скорости тактирования МК stm32f103c8t6 для кварцевого резонатора 8 МГц

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Рис. 18. Значения битов регистра DMA_ISR. Все биты доступны только для чтения. Значение после сброса: 0x0000 0000

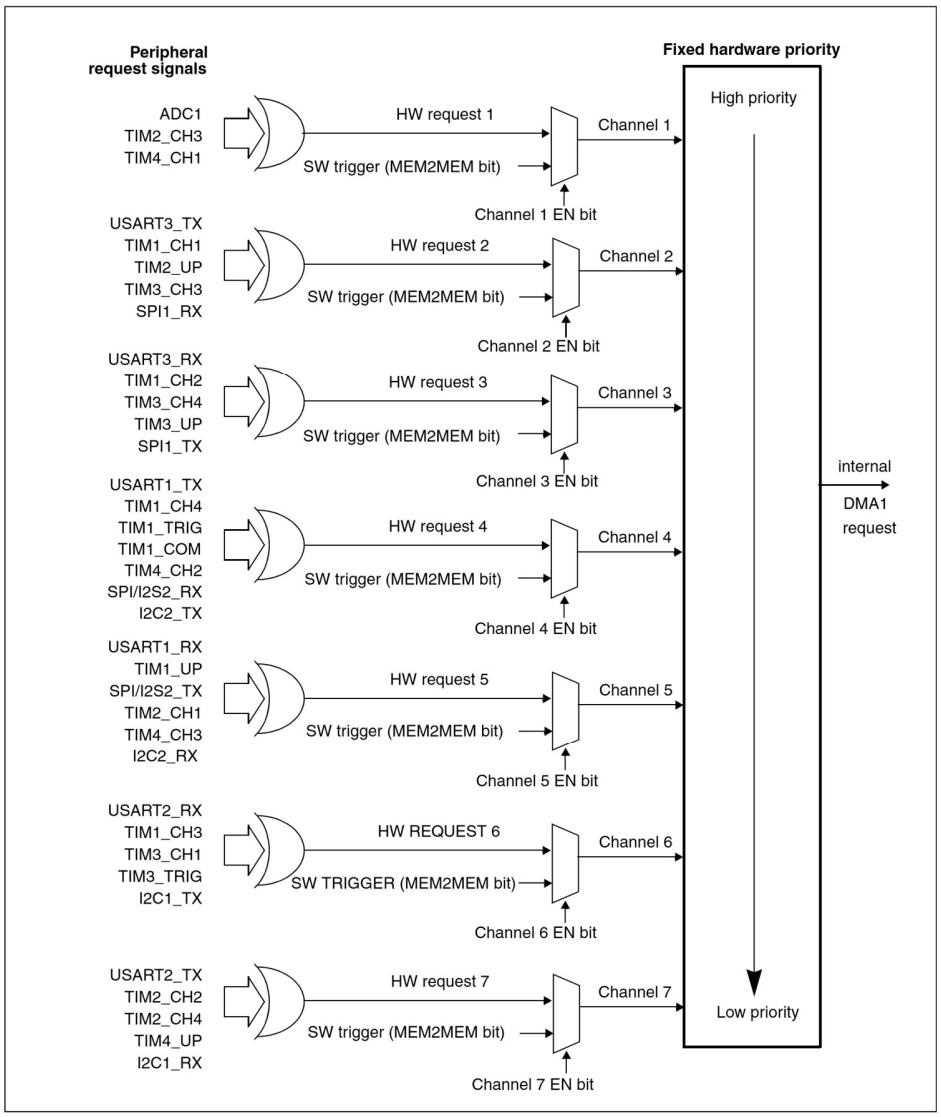

Рис. 19. Карта запросов блока DMA

Заполнения матрицы происходит по довольно простому алгоритму.

Алгоритм S (рис. 20**)**. (*Обновление отображаемых значений в виброматрице на сдвиговых регистрах*). На вход поступает сжатая до одного измерения карта глубины M размерами x, y. Для данного случая предполагается, что $xy = 8n$ где n – количество сдвиговых регистров. Пусть – размер входной карты, i – текущий регистр.

S1 [Все регистры обновлены?]. Получить следующее значение для матрицы. Если $i = l$, работа алгоритма заканчивается.

S2 [Передать значения регистру] Передать регистру значения с $8i$ по $8(i + 1) - 1$ (нумерация с 0).

S3 [Следующий регистр] Увеличить i.

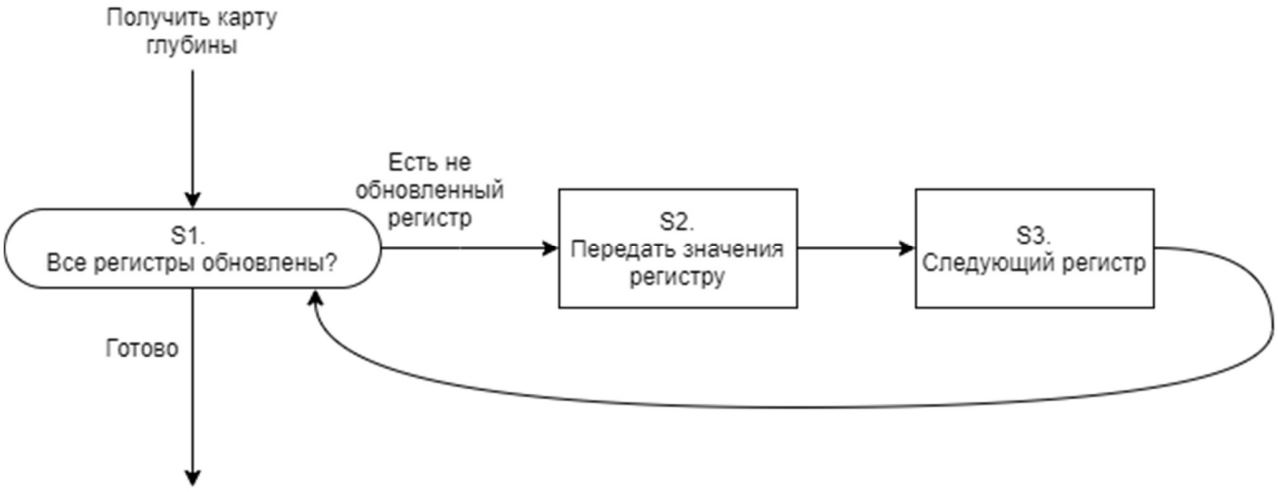

Рис. 20. Алгоритм S: обновление отображаемых значений в виброматрице на сдвиговых регистрах

2.2 Подключение вибромоторов

Для защиты выходов управляющих микросхем и возможности подключения более мощных вибромоторов, была применена пара Дарлингтона в составе схемы uln2803 [49] (рис. 21). Вход каждой пары подключается к выходу управляющей схемы, выход – к вибромотору. Данная микросхема состоит из 8 высокоточных NPN транзисторных пар Дарлингтона. Из рис. 22 видно, что каждая пара собрана по схеме с общим эмиттером, выход является открытым

коллектором. Также предусмотрены защитные диоды для индуктивных нагрузок. Согласно документации, входное напряжение для перехода во включенное состояние должно быть не меньше 2.4 V, что соответствует выходному напряжению всех рассмотренных решений (напряжение питания (5 В для прототипа) для sn74hc595, 0.7 V_{DD} для PCA9685 (где V_{DD} – напряжение питания, 5 В в прототипе. Выходное напряжение = 3.5 В)).

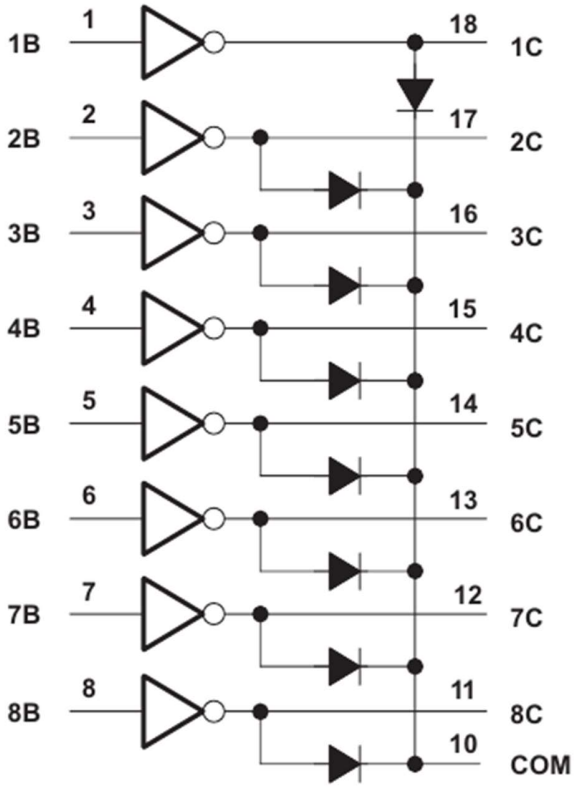

Рис. 21. Логическая диаграмма uln2803.

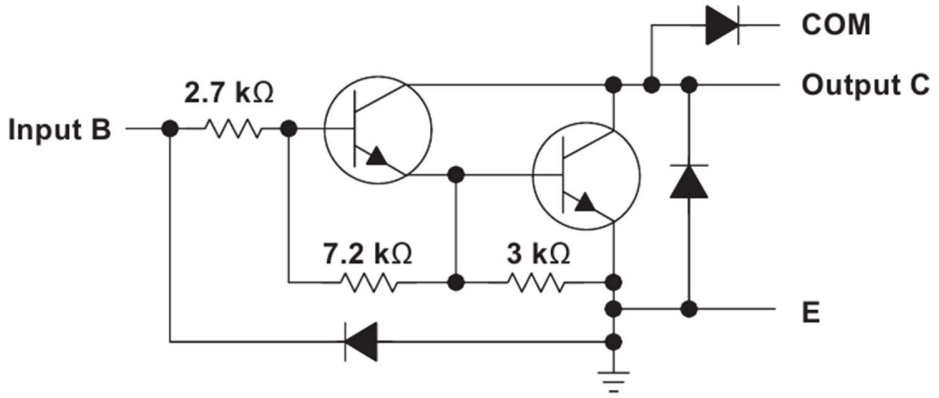

Рис. 22. Функциональная схема одной пары Дарлингтона микросхемы uln2803

Из рис. 23 видно, что временные характеристики удовлетворяют заданным ШИМ сигналам (для f_{PWM} = 1 кГц с разрядностью ШИМ 256 длительность одного разряда равна $t = \frac{1}{(f_{PWM} * n_{PWM})} = \frac{1}{(1000*256)} \approx 4$ мкс, что больше времени распространения сигнала).

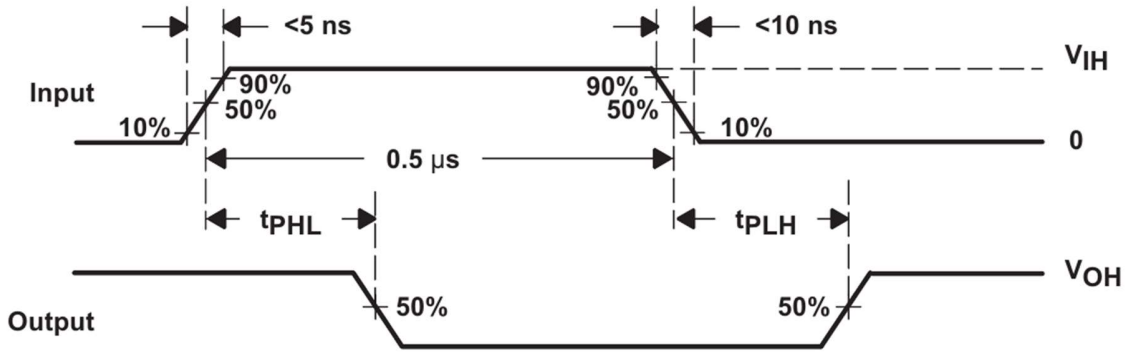

Рис. 23. Задержки распространения микросхемы uln2803, где t_{PLH} = 130 нс (переход из состояния выключено в состояние включено),
t_{PHL} = 20 нс (переход из состояния включено в состояние выключено).

2.3 Виброматрица на основе многоканальных ШИМ-контроллеров (PCA9685)

В качестве микросхем управления вибромоторами хорошо показали себя драйверы LED.

Для реализации данного решения можно использовать продемонстрированные ранее способы (рис. 12, рис. 13) подключения вибромоторов через сборки Дарлингтона, заменив сдвиговые регистры на ШИМ-контроллеры (рис. 24). На рис. 24 J1-J6 – выводы выбора адреса контроллера, J7 подключается к вибромоторам через сборки Дарлингтона.

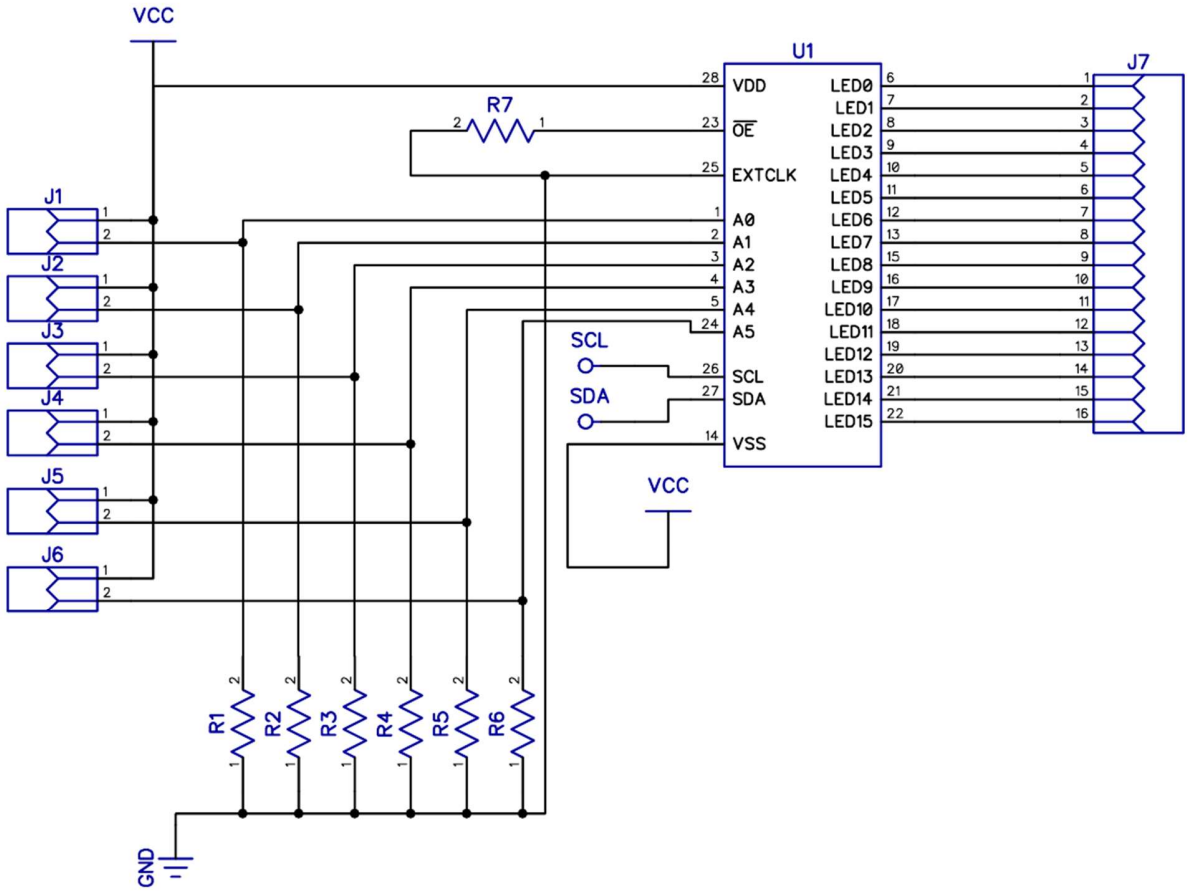

Рис. 24. Схема управления 16 вибромоторами при помощи PCA9685

PCA9685 [50] представляет из себя 16 канальный, 12 разрядный ШИМ контроллер с управлением по шине I2C. Каждый выход имеет собственный 12 разрядный ШИМ контроллер с фиксированной частотой. Частота задается программно и может варьироваться от 24 до 1256 Гц. Таким образом для всех

каналов ШИМ задается одна частота из этого диапазона. Скважность каждого выхода может изменяться от 0 % до 100 %.

После передачи START последовательности (начало передачи), мастер I^2C шины (bus-master) должен передать адрес ведомого, к которому ему необходимо получить доступ. Для возможности подключения нескольких ШИМ контроллеров к одной шине, в PCA9685 предусмотрена установка I^2C адреса ведомого (ШИМ контроллера) через программные выводы A0-A5 (рис. 25). Таким образом, существует максимум ▪ возможных адресов для установки, используя 6 аппаратных пинов. Два этих адреса, программный сброс (Software Reset, 0000 0110) и широковещательный адрес (All Call, 1110 000, 0x70) не могут быть использованы, оставляя тем самым 62 доступных адресов для микросхем. Адрес ведомого задается согласно рис. 26.

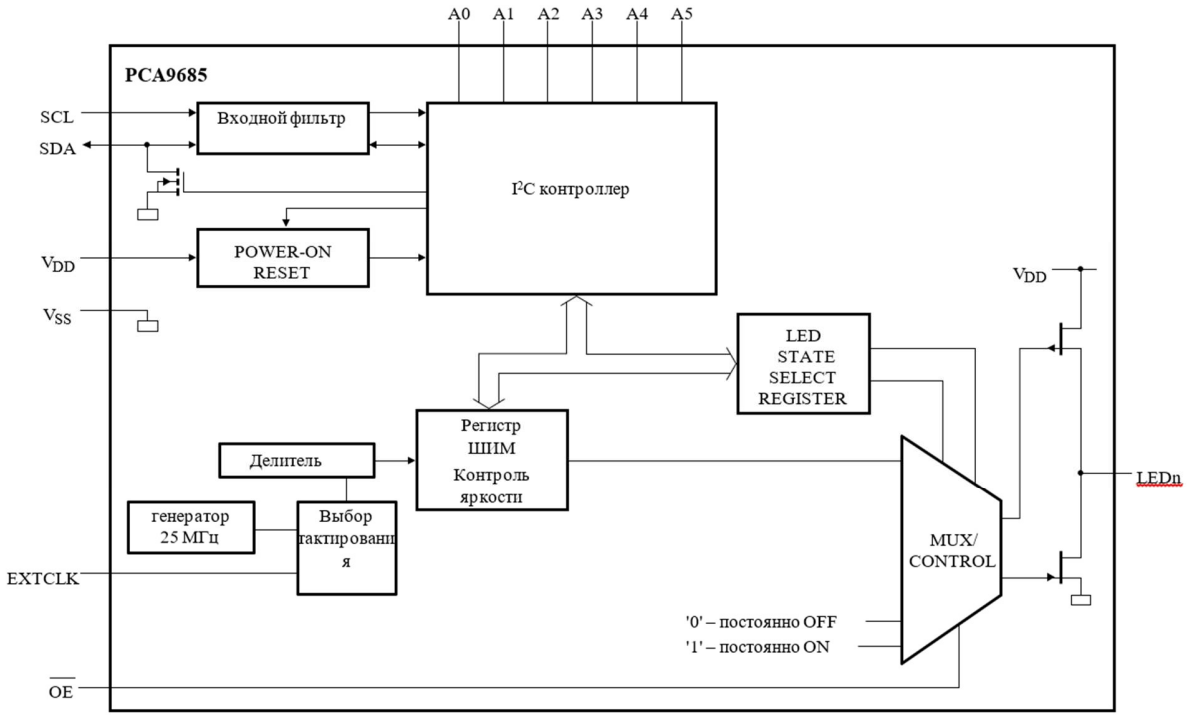

Рис. 25. Блок-схема PCA9685

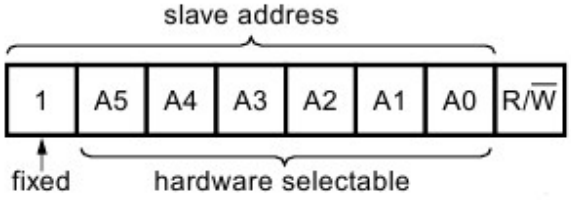

Рис. 26. Адрес ведомого (slave):
первый бит зафиксирован (1), далее идут 6 устанавливаемых аппаратно битов и флаг чтения/записи

Далее после выбора ведомого, ведущий должен передать байт ШИМ контроллеру, который будет храниться в регистре команд (рис. 27), который будет использоваться как указатель на регистр, к которому необходимо получить доступ.

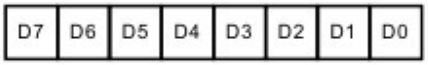

Рис. 27. Регистр команд, начальное значение = 0

PCA9685 возможно перевести в спящий режим (установка бита 4 MODE1(см. таблицу 3)) без остановки какого-либо из каналов PWM, бит RESTART (Состояние Restart) (бит 7 MODE1) будет установлен на логическую 1 в конце цикла обновления ШИМ. Содержимое каждого регистра ШИМ остается действительным, когда тактирование выключено.

Таблица 3. Описание регистра MODE1

** Значение по умолчанию.*

Бит	Символ	Доступ	Значение	Описание
7	RESTART	R		Показывает состояние RESTART.
		W		Запись 1 в этот бит очищает до логического состояния 0. Запись логического 0 не дает никакого эффекта.
			0*	Restart выключен.
			1	Restart включен.
6	EXTCLK	R/W		Для использования пина EXTCLK, этот бит должен быть установлен согласно следующей последовательности: 1. Установка бита SLEEP в MODE1. Это выключает внутренний генератор. 2. Запись логических единиц в биты SLEEP и EXTCLK в MODE1. Происходит переключение. Внешний сигнал тактирования может быть активен во время переключения, потому что бит SLEEP установлен. Это “sticky-бит”, то есть его нельзя сбросить, записав в него логический 0. Бит EXTCLK может быть сброшен только с помощью цикла включения питания или программного сброса. Диапазон EXTCLK составляет от 0 до 50 МГц. $$\text{время\_обновления} = \frac{\text{EXTCLK}}{4096 * (\text{делитель} + 1)}$$ Где: EXTCLK – входная частота
			0*	Использовать внутренний генератор.
			1	Использовать тактирование с пина EXTCLK.
5	AI	R/W	0*	Автоматическое увеличение регистров выключено.
			1	Автоматическое увеличение регистров ключено.
4	SLEEP	R/W	0	Нормальный режим
			1*	Режим пониженного энергопотребления. Генератор выключен
3	SUB1	R/W	0*	PCA9685 не отвечает на подадрес 1 шины I^2C.
			1	PCA9685 отвечает на подадрес 1 шины I^2C.
2	SUB2	R/W	0*	PCA9685 не отвечает на подадрес 2 шины I^2C..
			1	PCA9685 отвечает на подадрес 2 шины I^2C.
1	SUB3	R/W	0*	PCA9685 не отвечает на подадрес 3 шины I^2C.
			1	PCA9685 отвечает на подадрес 3 шины I^2C.
0	ALLCALL	R/W	0	PCA9685 не отвечает на адрес шины I^2C LED All Call.
			1*	PCA9685 отвечает на адрес шины I^2C LED All Call.

Чтобы перезапустить все ранее активные каналы ШИМ с помощью нескольких циклов шины I2C, необходимо выполнить следующие действия:

1. Чтение регистра MODE1.
2. Проверить, что 7-й бит установлен. Если это так, необходимо сбросить бит 4 (SLEEP).
3. Ожидание стабилизации генератора (500 мкс).
4. Запись бита в 7-й бит регистра MODE1. Все каналы ШИМ перезапустятся, и бит RESTART будет сброшен.

Дополнительная конфигурация выходов доступна в регистре MODE2 (см. таблицу 4).

Таблица 4. Описание регистра MODE2

Бит	Символ	Доступ	Значение	Описание
7 to 5	-	Только чтение	000*	зарезервировано
4	INVRT	R/W	0*	Выходное значение логически не инвертировано.
			1	Выходное значение логически инвертировано.
3	OCH	R/W	0*	Выходное значение меняется при команде STOP.
			1	Выходное значение меняется при ACK.
2	OUTDRV	R/W	0	Выходы 16 LEDn сконфигурированы как с открытым стоком.
			1*	Выходы 16 LEDn сконфигурированы как totem pole.
1 to 0	OUTNE[1:0]	R/W	00*	Если OE = 1 (выходные драйверы не включены), LEDn = 0.
			01	Если OE = 1 (выходные драйверы не включены), LEDn = 0: LEDn = 1 когда OUTDRV = 1 LEDn = высокий импеданс OUTDRV = 0 (такое же как OUTNE[1:0] = 10)
			1X	Когда OE = 1 (выходные драйверы не включены), LEDn = высокий импеданс.

Время включенного и выключенного состояния для каждого канала i определяется независимо в регистрах LEDi_ON и LEDi_OFF (см. таблицу 5). Для каждого выхода микросхемы есть два 12-битных регистра. Первый 12-битный регистр содержит значение времени включенного состояния, второй - значение времени выключенного состояния. Значения этих двух регистров сравниваются со значением 12 битного таймера, идущего непрерывно с 0 до 4095 (0xFFF).

Каждый регистр возможно программировать согласно таблице 5. Таким образом, возможно менять фазовый сдвиг с разрешением $\frac{1}{2^{12}} = \frac{1}{4096}$ от заданной частоты.

Таблица 5. Первые 22 регистров. Далее регистры LEDi_ON_L, LEDi_ON_H, LEDi_OFF_L, LEDi_OFF_H имеют аналогичное значение для каждого канала i, вплоть до LED15_OFF_H.

Регистр # (десятичная система счисления)	**Регистр # (шестнадцатеричная система счисления)**	**Бит 7**	**Бит 6**	**Бит 5**	**Бит 4**	**Бит 3**	**Бит 2**	**Бит 1**	**Бит 0**	**Название**	**Тип**	**Функция**
0	00	0	0	0	0	0	0	0	0	MODE1	чтение/ запись	Регистр mode 1
1	01	0	0	0	0	0	0	0	1	MODE2	чтение/запись	Регистр mode 2
2	02	0	0	0	0	0	0	1	0	SUBADR1	чтение/запись	Подадрес шины I^2C подадрес 1
3	03	0	0	0	0	0	0	1	1	SUBADR2	чтение/запись	Подадрес шины I^2C подадрес 2
4	04	0	0	0	0	0	1	0	0	SUBADR3	чтение/запись	Подадрес шины I^2C подадрес 3
5	05	0	0	0	0	0	1	0	1	ALLCALLADR	чтение/запись	Адрес шины I^2C LED All Call
6	06	0	0	0	0	0	1	1	0	LED0_ON_L	чтение/запись	Байт 0 контроля выхода и яркости LED0
7	07	0	0	0	0	0	1	1	1	LED0_ON_H	чтение/запись	Байт 1 контроля выхода и яркости LED0
8	08	0	0	0	0	1	0	0	0	LED0_OFF_L	чтение/запись	Байт 2 контроля выхода и яркости LED0
9	09	0	0	0	0	1	0	0	1	LED0_OFF_H	чтение/запись	Байт 3 контроля выхода и яркости LED0
10	0A	0	0	0	0	1	0	1	0	LED1_ON_L	чтение/запись	Байт 0 контроля выхода и яркости LED1
11	0B	0	0	0	0	1	0	1	1	LED1_ON_H	чтение/запись	Байт 1 контроля выхода и яркости LED1
12	0C	0	0	0	0	1	1	0	0	LED1_OFF_L	чтение/запись	Байт 2 контроля выхода и яркости LED1

Продолжение таблицы 5

13	0D	0	0	0	0	1	1	0	1	LED1_OFF_H	чтение/запись	Байт 3 контроля выхода и яркости LED1
14	0E	0	0	0	0	1	1	1	0	LED2_ON_L	чтение/запись	Байт 0 контроля выхода и яркости LED2
15	0F	0	0	0	0	1	1	1	1	LED2_ON_H	чтение/запись	Байт 1 контроля выхода и яркости LED2
16	10	0	0	0	1	0	0	0	0	LED2_OFF_L	чтение/запись	Байт 2 контроля выхода и яркости LED2
17	11	0	0	0	1	0	0	0	1	LED2_OFF_H	чтение/запись	Байт 3 контроля выхода и яркости LED2
18	12	0	0	0	1	0	0	1	0	LED3_ON_L	чтение/запись	Байт 0 контроля выхода и яркости LED3
19	13	0	0	0	1	0	0	1	1	LED3_ON_H	чтение/запись	Байт 1 контроля выхода и яркости LED3
20	14	0	0	0	1	0	1	0	0	LED3_OFF_L	чтение/запись	Байт 2 контроля выхода и яркости LED3
21	15	0	0	0	1	0	1	0	1	LED3_OFF_H	чтение/запись	Байт 3 контроля выхода и яркости LED3

Для рис. 28: время задержки = 10%, коэффициент заполнения = 20 %. Время включенного состояния = 20 %, выключенного состояния = 80 %. Тогда значение для регистра 4095*0.1 = 409 (таймер считает от 0 до 4095) 409 = 0000 0001 1001 1001. Таким образом в регистр LEDi_ON_H заносится значение 1 (0000 0001), в LEDi_ON_L - 153 (1001 1001 или 0x99). Для времени включенного состояния 20% = 4095*0.2 = 819. Тогда значение для LEDi_OFF равняется 819+409=1228 (0000 0100 1100 1100). LEDi_OFF_L = 204 (1100 1100, 0xCC) LEDi_OFF_H = 4 (100).

Рис. 28. Пример для LEDi_ON = 409 и LEDi_OFF = 1228

Выходная частота ШИМ задается делителем, значение которого можно менять в регистре PRE_SCALE (см. таблицу 6). Аппаратно, минимальное значение, которое можно загрузить в регистр PRE_SCALE равняется 3. Значение делителя определяется следующим уравнением:

$$n_{prescale} = \left\lfloor \frac{f_{osc\_clock}}{n_{cnt} * f_{PWM}} \right\rfloor - 1, \quad (6)$$

где: $n_{prescale}$ – значение делителя;

f_{osc_clock} – частота задающего генератора, 25 МГц;

f_{PWM} – выходная частота ШИМ;

n_{cnt} – период счета.

Для PCA9685 $n_{cnt} = 2^{12} = 4096$. Тогда из (6):

$$n_{prescale} = \left\lfloor \frac{f_{osc\_clock}}{4096 * f_{PWM}} \right\rfloor - 1$$

Для минимального значения PRE_SCALE = 3, выходная частота ШИМ равняется:

$$f_{PWM} = \frac{25000000}{4096 * (3 + 1)} = 1\,526 \text{ Гц}$$

Размер регистра PRE_SCALE равен 8 бит. Максимальное значение, которое можно установить в регистр равно 255. Таким образом, минимальное значение частоты:

$$f_{PWM} = \frac{25000000}{4096 * (255 + 1)} = 24 \text{ Гц}$$

Между последовательностями START и STOP число переданных байтов не ограничено. Для подтверждения приема после каждого бита переданной информации передается один бит подтверждения (рис. 29). Ведущий устанавливает высокий уровень и генерирует дополнительный импульс синхронизации для подтверждающего бита. Ведомый должен генерировать подтверждение после каждого принятого байта.

Подтверждающее устройство должно подтянуть вниз линию SDA во время подтверждающего импульса синхронизации.

Таблица 6.

Регистры задающие соответствующие значения в регистрах всех каналов, значение делителя и регистр для установки тестового режима

Регистр # (десятичная система счисления)	**Регистр # (шестнадцатеричная система счисления)**	**Бит 7**	**Бит 6**	**Бит 5**	**Бит 4**	**Бит 3**	**Бит 2**	**Бит 1**	**Бит 0**	**Название**	**Тип**	**Функция**
250	FA	1	1	1	1	1	0	1	0	ALL_LED_ON_L	чтение/запись нуля	Загрузить во все регистры LEDn_ON байт 0
251	FB	1	1	1	1	1	0	1	1	ALL_LED_ON_H	чтение/запись нуля	Загрузить во все регистры LEDn_ON байт 1
252	FC	1	1	1	1	1	1	0	0	ALL_LED_OFF_L	чтение/запись нуля	Загрузить во все регистры LEDn_OFF байт 0
253	FD	1	1	1	1	1	1	0	1	ALL_LED_OFF_H	чтение/запись нуля	Загрузить во все регистры LEDn_OFF байт 1
254	FE	1	1	1	1	1	1	1	0	PRE_SCALE	чтение/запись	Делитель для выходной частоты ШИМ
255	FF	1	1	1	1	1	1	1	1	TestMode	чтение/запись	Вход в тестовый режим

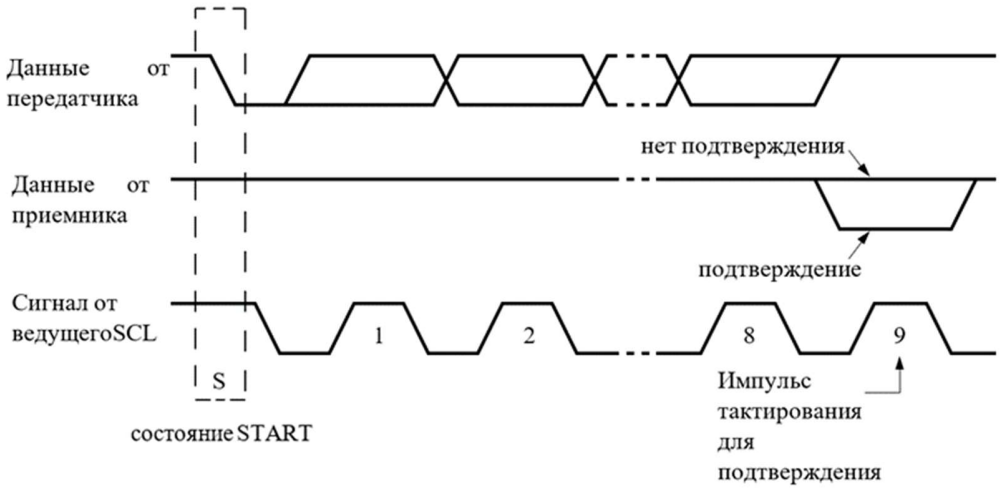

Рис. 29. Подтверждение на шине I2C

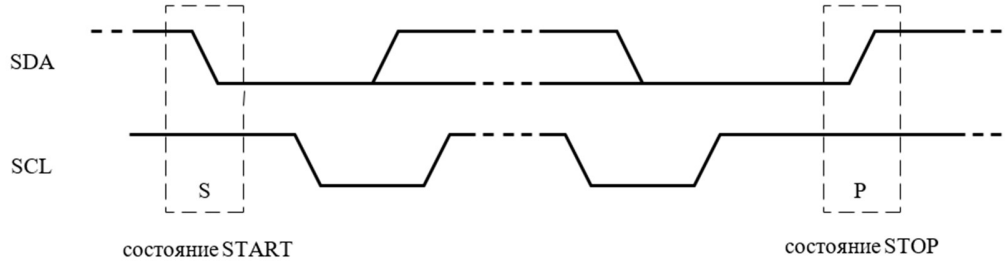

Рис. 30. Последовательности START и STOP

Алгоритм P (рис. 31**)**. (*Передача карты глубины ШИМ-контроллерам* PCA9685). На вход поступает сжатая до одного измерения карта глубины M размерами x, y. Для данного случая предполагается, что $xy = 16n$ где n – количество контроллеров PCA9685. Пусть l – адрес последнего контроллера, i – адрес текущего контроллера, j – текущий регистр (изначально $j = 6$), t – значение off_time.

P1 [Следующий контроллер?] Проверяется, остался ли контроллер, которому не было отправлено значение. Если $i = l$, работа алгоритма заканчивается.

P2 [Все регистры заполнены?] Проверяется, остался ли незаполненный регистр для данного контроллера. Если j = (16*4 + 6), выполняется переход к шагу P7.

P3 [Вычислить значения регистров] Для данного алгоритма время on_time равняется 0, значения выводов задает только регистр off_time. На данном шаге производится его вычисление (t).

P4 [Сформировать блок для отправки] Формируется последовательный блок данных для отправки. Для верхнего и нижнего регистров on_time задается нулевое значение, для двух регистров off_time производится вычисление нижнего (маска 0xFF) и верхнего значения (битовый сдвиг вправо на 8, $t \gg 8$)

P5 [Передать данные] Отправка блока по адресу контроллера i для адреса регистра j (регистры для установки off_time и on_time располагаются последовательно).

P6 [Установить следующий адрес регистра] Увеличить j на 4. Перейти к шагу P2.

P7 [Установить адрес следующего контроллера] Данный шаг зависит от конфигурации оборудования. Для прототипа адрес каждого следующего контроллера отличается на 1 и располагаются в неубывающем порядке. Таким образом, i увеличивается на 1.

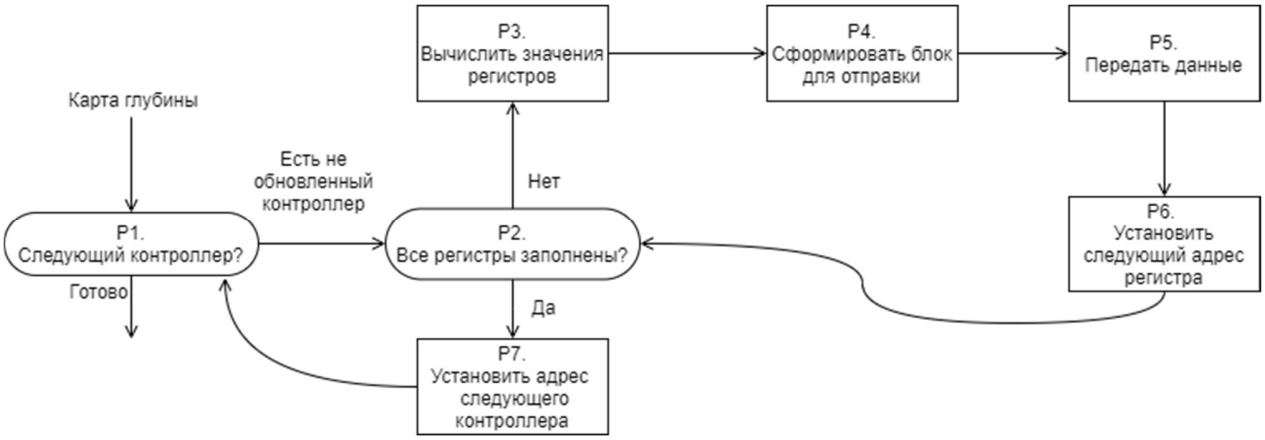

Рис. 31 Алгоритм P. (Передача карты глубины ШИМ-контроллерам PCA9685)

2.4 Использование одноплатных компьютеров для снятия и обработки изображений

В прототипе используется ПК, на котором запущена программа для снятия и обработки изображения. В дальнейшем он будет заменен на одноплатный компьютер наподобие Raspberry pi. Для примера в качестве двух плат для вычисления карты глубины были выбраны Raspberry IP 4 И Raspberry pi zero. Эти платы имеют разные процессоры с отличными микроархитектурами [51]. Raspberry IP 4 имеет BCM2711 (Broadcom) с ядром ARM A72 [52] (ARM v8). Raspberry pi zero - BCM2835 (Broadcom) с микроархитектурой ARM1176JZF-S (ARMv6). Помимо очевидных различий в тактовой частоте, BCM2711 имеет технологию Neon [53], которая является расширением SIMD (single instruction, multiple data — одиночный поток команд, множественный поток данных). Инструкции Neon могут выполнятся на нескольких элементах одновременно. Соответственно, с помощью этой технологии ускоряется обработка видео и аудио декодирование и кодирование, 2D/3D графика, а также алгоритмы обработки сигналов, компьютерного зрения и глубокого обучения. Таким образом, это технология расширяет архитектуру набора команд (ISA), предоставляя дополнительные инструкции, которые могут выполнять

математические операции на нескольких параллельных потоках данных. Это важно, так как при обработке больших объёмов данных, значительным фактором, уменьшающем производительность является время выполнения центральным процессором инструкций для обработки данных. Это время зависит от количества инструкций, которые необходимо выполнить с целым набором данных, а количество инструкций завесит от количества элементов данных, которые каждая инструкция способна обработать. Большинство инструкций ARM является SISD (Single Instruction, Single Data, Одиночный поток Команд, Одиночный поток Данных), каждая инструкция выполняет определенную операцию на единственном источнике данных. Таким образом, для обработки набора данных необходимо выполнить несколько инструкций. SIMD инструкции производят те же действия для нескольких единиц данных. Эти данные упакованы в более больших регистрах.

Таким образом, библиотеку OpenCV для BCM2711 (Raspberry IP 4) возможно скомпилировать с поддержкой VFPv3 и NEON. Это должно ускорить обработку изображений для получения карт глубины. Это делается добавлением конфигураций ENABLE_NEON=ON и ENABLE_VFPV3=ON в cmake.

Для профилирования программы вычисления карты глубины на двух вышеупомянутых платах был написан декоратор, который измеряет время выполнения функции (для данной программы метод compute). Далее на рис. 32 приведен код данного декоратора.

```
def time_taken(f, name=None):
    if name is None:
        name = f.__name__

    @wraps(f)
    def wrap(*args, **kwargs):
        start_time = perf_counter()
        res = f(*args, **kwargs)
        end_time = perf_counter()
        time_passed = end_time - start_time
        print(name, '\nTime taken:', time_passed, 's\n', str(timedelta(seconds=time_passed)))
        return res

    return wrap
```

Рис. 32. Код декоратора, измеряющий время выполнения функции

Здесь декоратор @wraps из стандартной библиотеки functools [54] сохраняет метаданные декорируемой функции. Далее замеряется время, которое будет потрачено на вызов принимаемой функции. Для этого используется perf_counter из стандартной библиотеки time [55]. Он возвращает значение (в секундах) счетчика производительности (performance counter). Данный счетчик предназначен для измерения коротких интервалов и обладает самым высоким разрешением из доступных. Функция timedelta из datetime [56] используется для более наглядного вывода информации о времени, затраченном на вычисление переданной функции. Она возвращает переданное время в формате hh:mm:ss.s (TIMEw.d).

На рис. 33 показано время вычисления карты глубины для Raspberry IP 4 2G, на рис. 34 – для pi zero w для тех же условий.

```
(cv) pi@raspberrypi:~/time_test $ python main.py
compute
Time taken: 0.0443463450001218 s
 0:00:00.044346
```

Рис. 33. Время вычисления карты глубины для Raspberry IP 4 2G

```
(cv) pi@raspberrypi:~/cv_time_test $ python3 main.py
compute
Time taken: 0.7400557529999787 s
 0:00:00.740056
```

Рис. 34. Время вычисления карты глубины для Raspberry IP zero

Таким образом, старшая модель для одинакового изображения вычисляет карту глубины в 16 раз быстрее, чем младшая. Однако для младшей модели время вычисления карты глубины является приемлемым, она способна обеспечивать обновление значений вибрoматрицы примерно раз в секунду.

2.5 Построение карты глубины

В качестве алгоритмов построения кары глубины были использованы ранее упомянутые SGBM, BM и FCRN.

Для сравнительных тестов будет использоваться изображение, которое показано на рис. 35 [57]. На рис. 36 изображено истинная карта глубины (ground truth).

Рис. 35. Фото, снятое с левой камеры

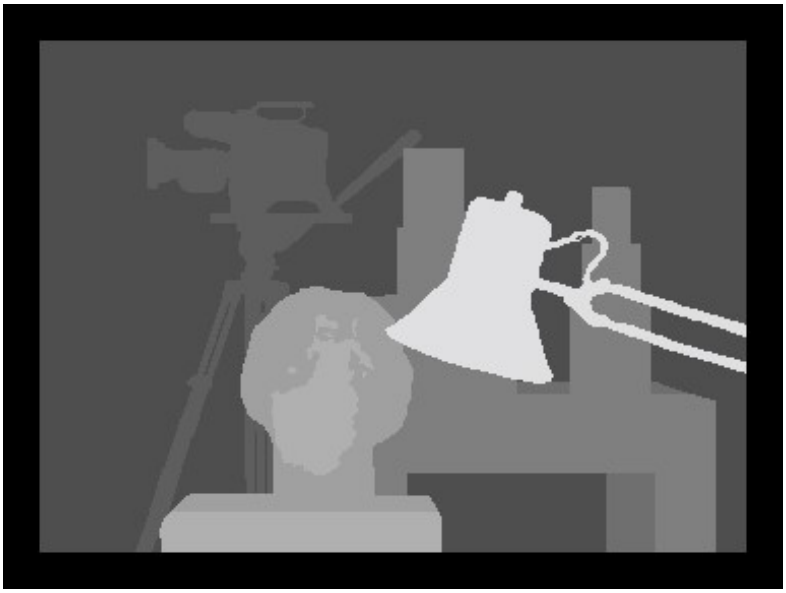

Рис. 36. Истинная карта глубины (ground truth)

Программная реализация алгоритма SGBM была взята из библиотеки opencv версии 4.5.1. Приложение вычисления карты глубины было написано на Python. Для заданного изображения были найдены значения numDisparities (максимальная диспаратность минус минимальная диспаратность) и blockSize. Они передаются функции StereoSGBM_create. Их выбор осуществлялся на основании рекомендаций документации и сравнения результата с истинной картой глубины.

Всего для функции StereoSGBM_create предусмотрено 11 необязательных параметров [58] (рис. 37)

```
int minDisparity = 0,
int numDisparities = 16,
int blockSize = 3,
int P1 = 0,
int P2 = 0,
int disp12MaxDiff = 0,
int preFilterCap = 0,
int uniquenessRatio = 0,
int speckleWindowSize = 0,
int speckleRange = 0,
int mode = StereoSGBM::MODE_SGBM
```

Рис. 37. Параметры StereoSGBM

Где: minDisparity – минимально возможное значение диспаратности. Обычно оно равно нулю, но иногда алгоритмы ректификации могут сдвигать изображения, поэтому этот параметр необходимо соответствующим образом отрегулировать.

numDisparities – максимальное значение диспаратности минус минимальная диспаратность. Значение всегда больше нуля.

blockSize – минимальный размер блока. Обычно он должен находиться в диапазоне 3÷11.

P1 – параметр, контролирующий сглаживание диспаратности.

P2 - параметр, контролирующий сглаживание диспаратности. Чем больше этот параметр, тем более слаженнее диспаратность. P1 это штраф на изменение диспаратности с минус или минус 1 между соседними пикселями. P2 – штраф на изменение диспаратности на более чем 1 между соседними пикселей. Необходимо, чтобы P2 > P1.

disp12MaxDiff – максимальная разрешенная разница (в целых пикселях) при проверке диспаратности между левым и правым значение

preFilterCap – усечение значения пикселей изображения до фильтра. Вычисляется производная по x и для каждого пикселя оно обрезается до интервала [-preFilterCap, preFilterCap].

uniquenessRatio – разница в процентах, при которой наилучшее значение вычисленной функции стоимости должно «выиграть» второе лучшее значение, чтобы найденное совпадение считалось правильным.

speckleWindowSize – максимальный размер областей сглаживания диспаратности, с учетом их шумовых зернистостей и исключения их.

speckleRange – максимальная вариация диспаратности в каждой соединенной компоненте.

mode – при установке StereoSGBM::MODE_HH выполняется полноразмерный двухпроходный алгоритм динамического программирования. Он занимает O(W*H*numDisparities) байт.

Код создания карты глубины показан на рис. 38

```
stereo = cv2.StereoSGBM_create(numDisparities=16, blockSize=15)
disparity = stereo.compute(imgL, imgR)
disparity = disparity[:, 17:]
disparity_cub = cv2.resize(disparity, (6, 6))
```

Рис. 38. Код создания карты глубины

Где: imgL – левое изображение;

imgR – правое изображение.

Далее переменные disparity и disparity_cub являются картами глубины размера, близкого к оригинальному (рис. 39) и масштабирования до размера 6*6 (рис. 40). Размер карты 6*6 выбран для матрицы 6*6 и предназначен для передачи виброматрице. В нем значение каждого пикселя отражает расстояние до объекта.

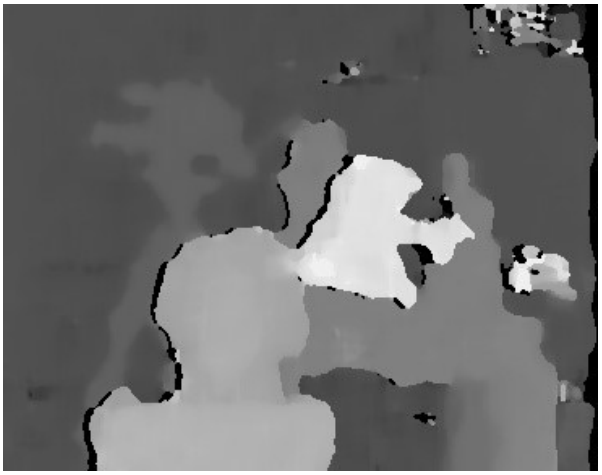

Рис. 39. Карта глубины, полученная для SGBM. Размер, близок к оригинальному

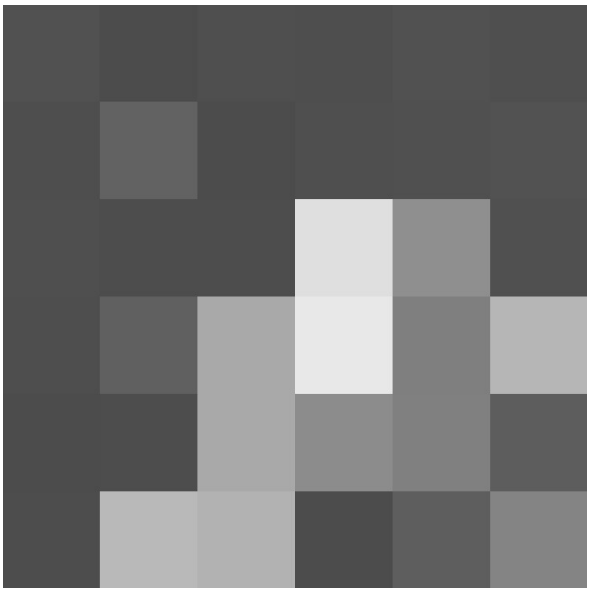

Рис. 40. Масштабированная до 6*6 пикселя карта глубины (увеличенное изображение)

Алгоритм BM также был взят из библиотеки opencv. Функция создания объекта StereoBM [18], StereoBM_create, принимает 2 параметра:

- numDisparities - диапазон поиска диспаратности. Для каждого пикселя алгоритм найдет наилучшую диспаратность от 0 (минимальная диспаратность по умолчанию) до numDisparities. Диапазон поиска можно сместить, изменив минимальную диспаратность.

- blockSize - линейный размер блоков, сравниваемых алгоритмом. Размер должен быть нечетным (так как блок центрируется в текущем пикселе). Больший размер блока подразумевает более гладкую, но менее точную карту диспаратности. Меньший размер блока дает более подробную карту диспаратности, но у алгоритма больше шансов выдать неправильный результат.

Остальные параметры устанавливаются (PreFilterCap, TextureThreshold и тд) устанавливаются через функции.

Для настройки параметров использовался графический интерфейс [59], позволяющая визуально выбрать подходящие параметры для алгоритма. Результат показан на рис. 41. Как видно, на карте глубины присутствуют большее количество шумов, чем на карте, построенной при помощи предыдущего алгоритма.

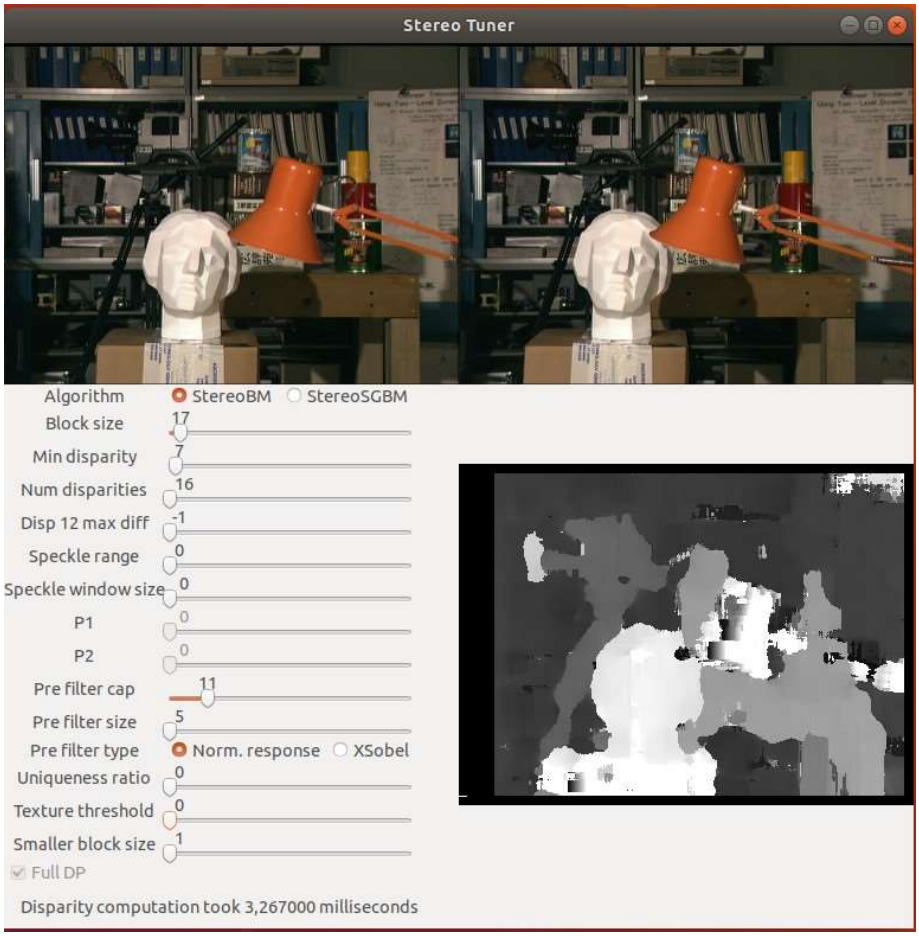

Рис. 41. Параметры для BM

Для построения карты глубины на основе одного изображения с помощью FCRN была использована работа [60]. В данном репозитории содержится обученная CNN модель, обученная для вычисления карты глубины из единичного входного RGB изображения. Сеть была построена согласно работе [35]. Предоставленные модели использовались для получения результатов, представленных в работе [35], по тестовым наборам данных NYU Depth v2 и Make3D для сцен в помещении и на улице соответственно. Таким образом, данные модели возможно использовать для произвольных изображений для получения карты глубины. Для простоты дальнейшего портирования для работы с моделью использовался фреймворк TensorFlow. Полученная карта глубины для левого изображения показана на рис. 42.

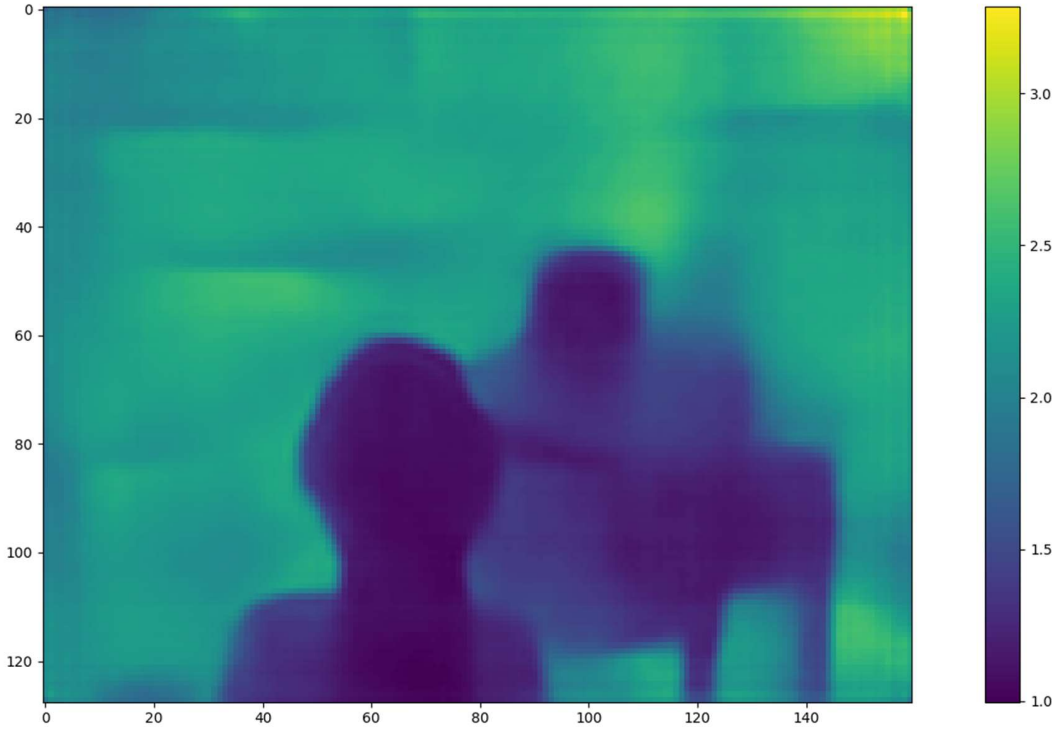

Рис. 42. Результат работы FCRN для левого изображения (Рис. 35)

Можно заметить, что карта глубины, полученная с использованием одной камеры (рис. 42) в сравнении с результатом, полученным для двух камер (рис. 39), имеет меньше деталей как на переднем, так и на заднем фоне. Однако при невозможности использования двух камер, результат, полученный FCRN, является удовлетворительным. Однако в данном проекте имелась возможность использования двух камер, а достаточно большой размер модели FCRN и относительно ресурсозатратное вычисление карты глубины ограничивает использование этого метода в оборудовании, где необходима мобильность.

2.6 Передача карты глубины в виброматрицу

Как отмечалось выше, вычисленная карта глубины далее передается в виброматрицу, где отображается в виде интенсивности вибрации вибромоторов. В данной работе применялся преимущественно протокол SPI и I2C.

В обоих случаях построения карты двумерный массив, возвращенный функцией opencv resize, необходимо преобразовать в одномерный массив для последовательной передачи принимающим устройствам. Для этого данный массив сжимается до одного измерения (функция ndarray.flatten из библиотеки numpy) и передается по выбранному протоколу матрице.

В случае с Raspberry pi за передачу по SPI отвечают пины GPIO 12,13,14 (рис. 43) [61]. Согласно документации на периферию BCM2711 [62], частота тактирования SPI определяется по формуле:

$$SPIx\_CLK = \frac{system\_clock\_freq}{2*(speed\_field+1)},$$

где: System_clock_freq – частота тактирования системы (250 МГц);

Speed_field – поле, задающее деление системной частоты для SPI.

Таким образом, для частоты тактирования системы 250 МГц и поля деления частоты равным нулю, частота SPI составляет 125 МГц. Это максимальная теоретическая частота SIP для BCM2711. На практике частоту SPI необходимо выбирать ниже [62] так как пины ввода вывода не способны передавать или принимать сигналы на такой высокой скорости. Минимальная частота SPI составляет 30.4 КГц. Кроме того, для данного чипа присутствует возможность перед нормальным завершением цикла передачи установить время удержания длительностью полу-бит (рис. 44), в течении которого уровень тактирования не меняется, но остается частью цикла передачи/приема. Это делает гарантированным то, что любые данные MISO имеют как минимум время одного бита SPI для приема. Это делается при помощи системного тактирования. Для 250 МГц системное тактирование добавляет время удержания размерностью в 6 нс.

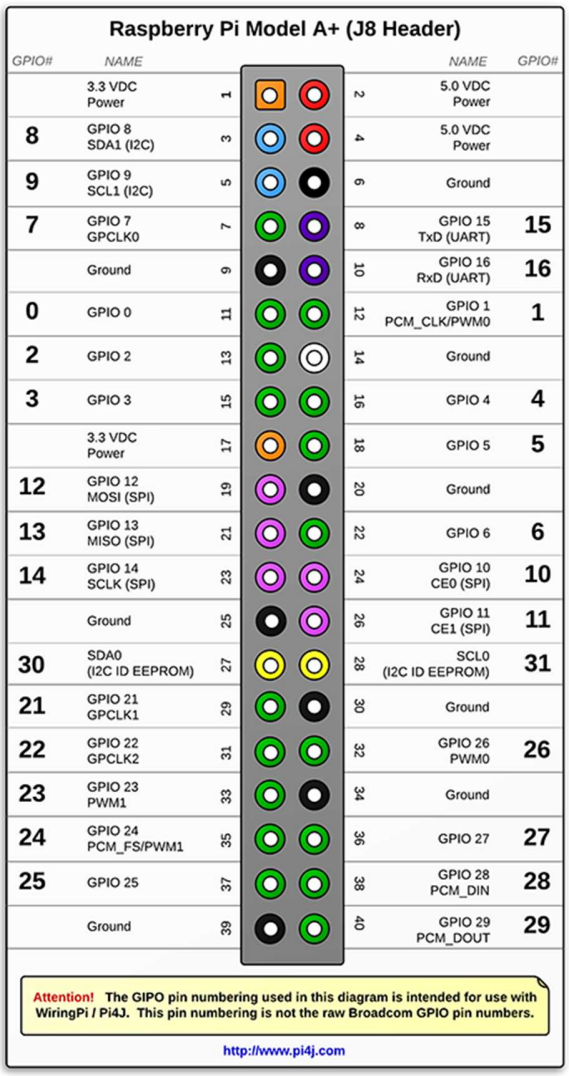

Рис. 43. Распиновка J8 (Raspberry pi 4 и Raspberry pi zero)

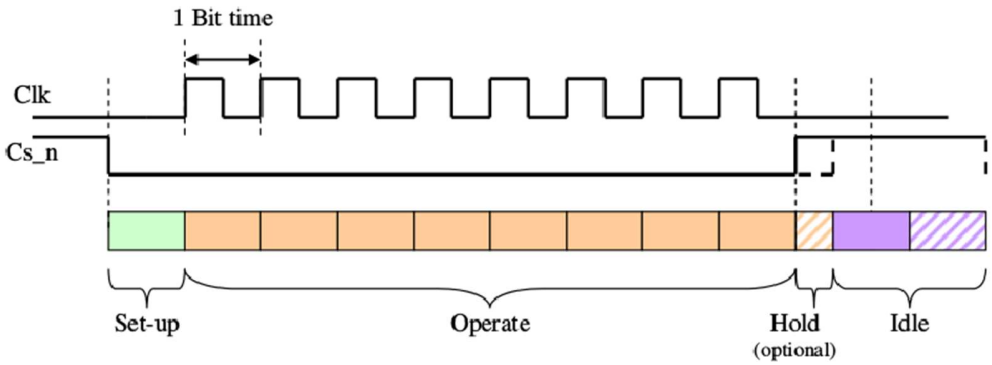

Рис. 44. Цикл доступа SPI

Для управления виброматрицей, построенной на PCA9685 в BCM2711 имеются BSC (Broadcom Serial Contro) контроллер, который является ведущим, работающим в fast-mode (400Kb/s) контроллером BSC. BSC – проприетарная шина, совместимая с шиной Philips® I2C.

Для передачи по I2C (PCA9685) используя аппаратные средства Raspberry pi была использована библиотека smbus2, которая в свою очередь является реализацией библиотеки python-smbus.

На рис. 45 представлена реализация алгоритма отправки карты глубины для PCA9685. Используется шина I2C 1. Предварительно была вычислена карта глубины, масштабирована до необходимого размера и сжата до одного измерения (переменная flt_data). Начальный адрес хранится в переменной init_adr и равен 0x40 (64) для случая, когда не задействуются перемычки смены адреса I2C. Далее происходит итерация по числу PCA9685, которое находится в виброматрице (переменная NUMBER_OF_PCA). Каждой микросхема PCA9685 в итерации первым делом инициализируется. Для этого по адресу PCA9685 отправляется бит 0x21 (10 0001) со смещением 0 (метод write_byte_data SMBus). Далее производится итерация по каждому регистру PCA9685. Разрешение одной микросхемы хранится в NUMBER_OF_PCA и равняется 16.

```
flt_data=list(disparity_cub.flatten())
with SMBus(1) as bus:
    addr=init_adr
    for i in range(NUMBER_OF_PCA):
        bus.write_byte_data(addr, 0, init_reg[0])
        for imod in range(PCA_RESOL):
            now_reg = 6 + 4 * imod
            snt_data=list()
            off_time=0
            off_time=math.ceil((flt_data[i*PCA_RESOL+imod]/DM_MAX_VAL)*PCA_MAX_VAL)
            snt_data=[0x0, 0x0, int(off_time & 0xFF), int((off_time >> 8) & 0xF)]
            bus.write_i2c_block_data(addr, now_reg, snt_data)
        addr+=1
```

Рис. 45. Передача карты глубины PCA9685, используя smbus2 (Raspberry pi)

Как было указано выше, для каждого вывода используются 4 регистра, сгруппированные как LEDi_ON_L, LEDi_ON_H, LEDi_OFF_L, LEDi_OFF_H для каждого выхода $0 \leq I \leq 15$. Для этого берется шаг 4 (4 * imod) согласно таблице 5 первые 6 регистров – служебные. Для этого вводится смещение на 6. Время, в течение которого данный выход будет выключен, определяется переменной off_time. Эта переменная вычисляется из самой карты глубины. Выбранное значение нормализуется по максимальному значению карты глубины и приводится относительно максимального возможного значения для одного выхода PCA (переменная PCA_MAX_VAL), которое равно (4095 (2^{12})). Полученное значение округляется путем отбрасывания дробной части. Далее формируется посылка, которая задает время выключенного состояния выхода. В данном случае применяются только регистры LEDi_OFF PCA9685. Для разбиения значения off_time на два регистра по байту в соответствии с документацией, применяется битовый сдвиг значения High, а также применение битовых масок для отсечения избыточной информации на один регистр и соответственно приведению числа к 8 битному размеру. Таким образом, переменная off_time контролирует скважность импульса заданного выхода. Как

можно видеть из примеров расчетов, это соответствует отсутствию задержки перед подачей высокого логического уровня на выход PCA9685 при новом начале цикла. Полученная посылка (snt_data) отправляется (метод write_i2c_block_data) с соответствующим данному регистру сдвигом. При полном заполнении одной микросхемы данными адрес увеличивается на 1. Т.е. в этом коде подразумевается, что в каждой следующей микросхеме PCA9685 задан адрес, на единицу больше предыдущего. Таким образом карта глубины распределяется по всем элементам виброматрицы.

2.7 Подключение виброматрицы к ПК по USB

В случае, когда для построения карты глубины используется ПК, карта глубины передается по порту USB компьютера. При этом для ее приема и отображения виброматрицей необходимо преобразовать этот пакет в воспринимаемый матрицей вид. Для этого в прототипе использовался микроконтроллер stm32f103.

В контроллерах stm32f103 имеется блок USB, совместимый со стандартом USB 2.0 full-speed 12 Мбит/с [47]. Периферийный блок USB реализует интерфейс между USB и шиной APB1 (рис. 15, рис. 46). Передача данных между ПК (хост) и системной памятью микроконтроллера происходит через выделенный пакетный буфер [48], доступ к которому получает непосредственно периферийный блок USB. Выделенная память имеет размер 512 байт. МК осуществляет детектирование токен-пакетов, управление приемом/передачей и обработку подтверждений (handshake), как этого требует стандарт USB. Формирование транзакций также осуществляется блоком USB (генерация контрольной суммы и проверка). Каждая конечная точка (endpoint) ассоциирована с блоком дескриптора буфера, который показывает, где находится память, выделенная для данной конечной точки, размер данной области или количество переданный байт. Когда маркер для действительной пары

функция/конечная точка распознается периферийным устройством USB, соответствующая передача данных (если требуется и если конечная точка настроена) происходит. Данные буферизируется периферийным USB-устройством, загружается во внутренний 16-битный регистр и доступ к памяти выделенный буфер выполняется. Когда все данные будут переданы, при необходимости, правильный пакет подтверждения через USB генерируется или ожидается в зависимости от направления передачи.

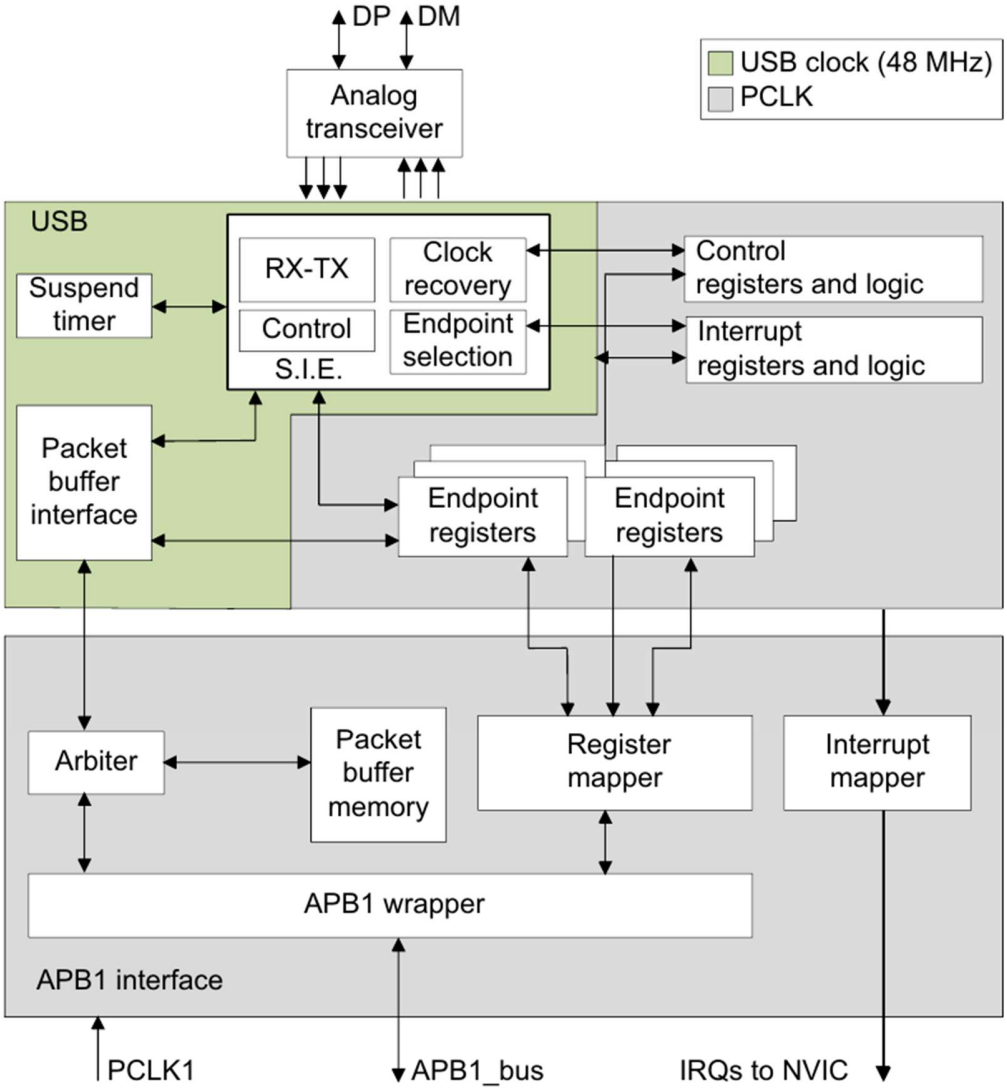

Рис. 46. Функциональная блок-схема блока USB

Блоки интерфейса USB МК:

- Serial Interface Engine (SIE) – данный блок отвечает за синхронизацию, распознавание паттернов, стаффинг битов, генерирование контрольных сумм и их проверка, верификация и генерация PID, оценка подтверждений. Он должен взаимодействовать с трансиверами USB и использовать виртуальные буферы, предоставленные интерфейсом пакетных буферов для хранения данных. Этот блок как же генерирует сигналы согласно событиям, происходящем в блоке USB.
- Timer – данный блок генерирует блокирующий start-of-frame импульс и детектирует состояние глобальной приостановки, когда трафик не получался в течении 3 мс.
- Packet Buffer Interface – управляет реализацией набора буферов памяти (для приема и передачи). Выбирает подходящий буфер согласно приходящим запросам от SIE и выделяет их в адресах памяти, которые указаны регистрами конечной точки. Увеличивает адрес после каждого слова до конца пакета, отслеживая число обменянных байтов и защищая буферы при достижении максимального размера.
- Endpoint-Related Registers – каждая конечная тока имеет соответствующий регистр, содержащий тип и текущий статус конечной точки. При однонаправленных конечных точках может быть использован единичный регистр для реализации двух отличных конечных точек. Количество регистров – 8. Это позволяет реализовать до 16 однонаправленных или 7 двунаправленных (Конечная точка 0 всегда используется для передачи управления в режиме с одним буфером) конечных точек.
- Control Registers – регистры, содержащие информацию о статусе блока USB и использующиеся для принуждения исполнения определенных действий USB (переход в выключенный режим, выход из него).

- Interrupt Registers – регистры, содержащие маски прерываний записи событий. Их можно использовать для выяснения причины прерывания, статуса прерывания или для очистки статуса ожидающего прерывания.
- Packet Memory – Это локальная память, которая физически содержит буферы пакетов. Его можно использовать в интерфейсе буфера пакетов, который создает структуру данных и может быть доступен непосредственно из прикладного программного обеспечения. Размер пакетной памяти составляет 512 байт, структурированных как 256 слов по 16 бит.
- Arbiter – Этот блок принимает запросы к памяти, поступающие от шины APB1 и от интерфейса USB. Он разрешает конфликты, отдавая приоритет доступу APB1, всегда резервируя половину пропускной способности памяти для завершения всех передач USB. Эта схема временного дуплекса реализует виртуальную двухпортовую SRAM, которая обеспечивает доступ к памяти во время выполнения транзакции USB. По этой схеме также разрешены многословные передачи APB1 любой длины.
- Register Mapper – Этот блок собирает различные байтовые и битовые регистры периферийного устройства USB в структурированный 16-битный набор слов, адресованный APB1.
- APB1 Wrapper – обеспечивает интерфейс с APB1 для памяти и регистров. Он также отображает всю периферию USB в адресном пространстве APB1.
- Interrupt Mapper – этот блок используется для выбора того, как возможные события USB могут генерировать прерывания и отображать их на три разные линии NVIC(USB low-priority interrupt - Прерывание USB с низким приоритетом (канал 20): инициируется всеми событиями USB (правильная передача, сброс USB), USB high-priority interrupt - прерывание USB с высоким приоритетом (канал 19): запускается только при правильном

событии передачи для изохронной массовой передачи и передачи данных с двойным буфером для достижения максимально возможной скорости передачи, USB wakeup interrupt - прерывание при выходе из спящего режима (канал 42): инициируется событием выхода из режима ожидания USB.).

Для приема данных по USB через виртуальный COM порт использовалась поставляемая вместе с CubeMX библиотека cube-usb-cdc [63]. Прием осуществляется через функцию CDC_Receive_FS файлausbd_cdc_if . Функция имеет следующие параметры:

```
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
```

где Buf – указатель на буфер приема;

Len – указатель на переменную, содержащую размер буфера.

Для прототипа, где карта глубины вычисляется на ПК, использовалась матрица на основе сдвиговых регистров. Соответственно, для передачи данных от микроконтроллера к матрице использовался протокол SPI.

2.8 Передача данных матрице по SPI

Микроконтроллер, использующийся в прототипе, имеет модули SPI [47] [48], которые при конфигурировании совместимы со сдвиговыми регистрами.

Согласно рис. 47 модуль SPI имеет 4 настраиваемых пинов: MISO, MOSI, SCK и NSS. NSS – выбор ведомого (slave) это необязательный пин для выбора ведомого устройства. Этот вывод действует как chip select(CS), позволяя ведущему (master) SPI обмениваться данными с ведомыми устройствами индивидуально для избегания конфликтов на линиях данных. Входы NSS ведомого могут управляться стандартными портами ввода/вывода на главном устройстве. Вывод NSS также может использоваться как выход, если он включен (бит SSOE), и быть выключенным, если SPI находится в ведущей конфигурации. Таким образом, все выводы NSS от устройств, подключенных к выводу ведущего

NSS, видят низкий уровень и становятся ведомыми (slaves), когда они настроены в аппаратном режиме NSS. При настройке в ведущем режиме с NSS, настроенным как вход (биты MSTR = 1 и SSOE = 0), и если на NSS устанавливается низкий уровень, SPI переходит в состояние сбоя ведущего режима: бит MSTR автоматически сбрасывается, и устройство настраивается в ведомый режим.

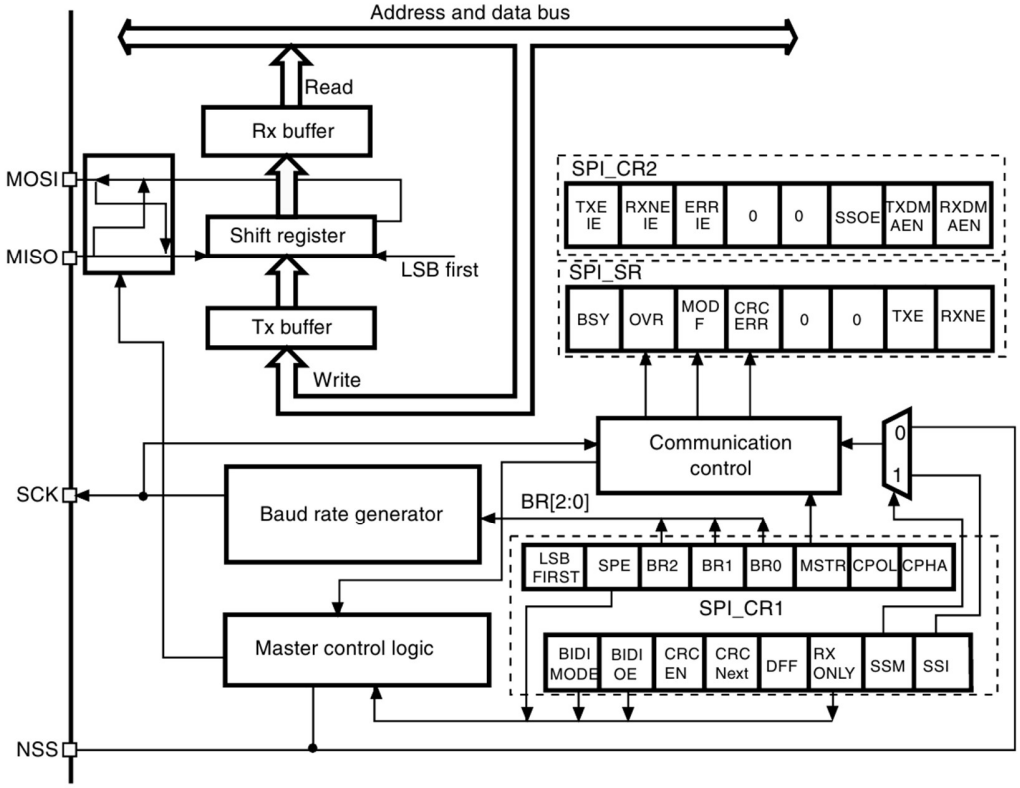

Рис. 47. Блок-диаграмма блока SPI

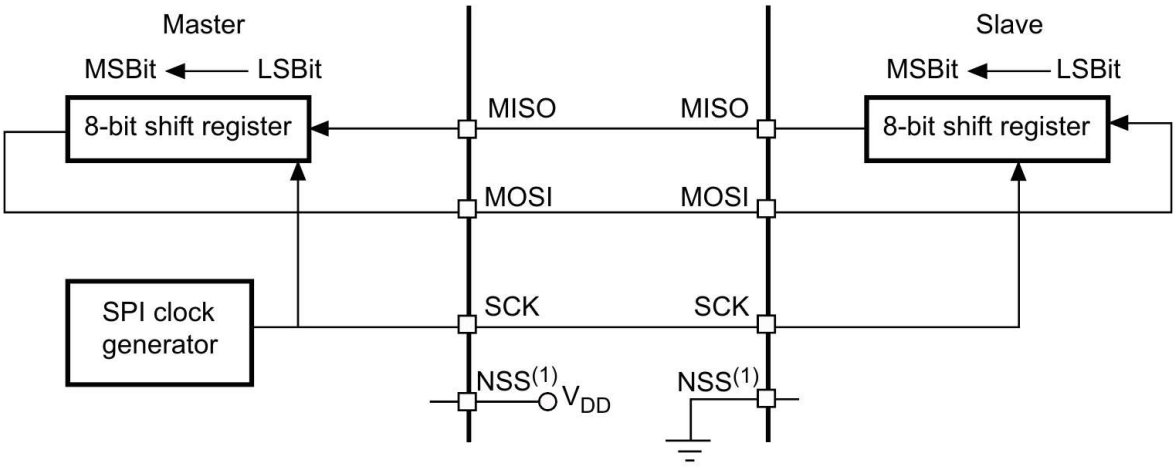

Рис. 48. Передача данных по SPI при одном мастере

Согласно рис. 14 и рис. 16, рис. 17, скорость передачи была выбрана 18 Мбит/с. Для установки полярности тактирования и фазы тактирования программно можно выбрать четыре возможных временных отношения, используя биты CPOL (полярность тактового сигнала) и CPHA (фаза тактового сигнала) в регистре SPI_CR1. Бит CPOL контролирует установившееся значение тактового сигнала, когда данные не передаются. Этот бит влияет как на ведущий, так и на ведомый режимы. Если бит CPOL сброшен, вывод SCK находится в низкоуровневом состоянии ожидания. Если установлен CPOL, вывод SCK находится в состоянии ожидания высокого уровня (рис. 49). Если бит CPHA установлен, второй фронт на выводе SCK (задний фронт, если бит CPOL сброшен, передний фронт, если бит CPOL установлен), является стробом захвата MSBit. Данные фиксируются при наступлении второго тактового перехода. Если бит CPHA сброшен, первый фронт на выводе SCK (задний фронт, если бит CPOL установлен, передний фронт, если бит CPOL сброшен), является стробом захвата MSBit. Данные фиксируются при возникновении первого тактового перехода. Комбинация битов CPOL (полярность тактового сигнала) и CPHA (фаза тактового сигнала) выбирает фронты захвата данных тактового сигнала. Для успешного приема отправленных данных sn74hc595 [44] (рис. 51) необходимо

выбрать низкую полярность и фазу, при которой первый фронт тактирования является первым фронтом захвата данных. Т.е. для регистра SPI_CR1 необходимо установить бит 1 (CPOL) равным нулю и бит 0 (CPHA) равным 0.

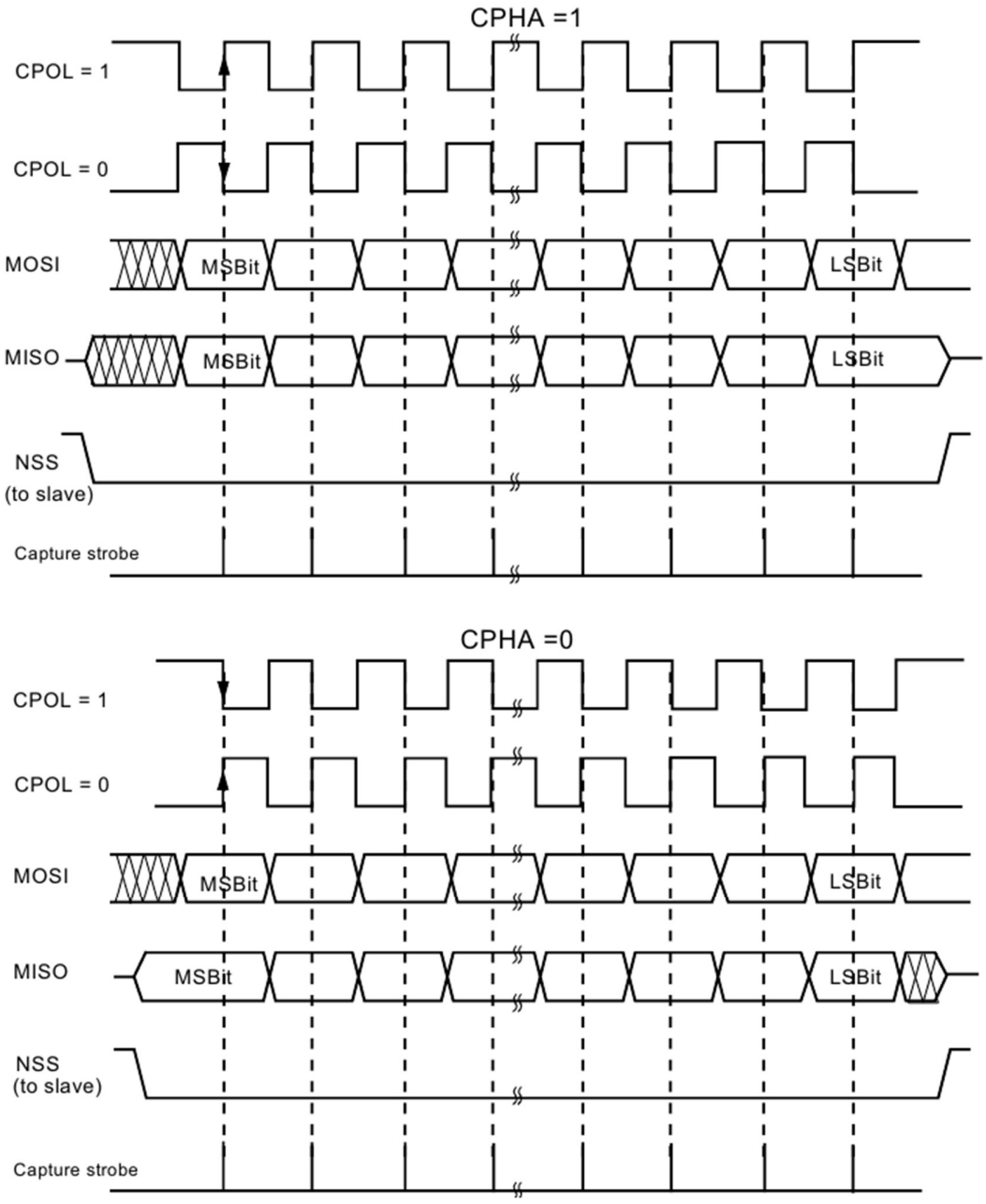

Рис. 49. Временная диаграмма передачи данных по SPI для STM32f103

Формат фрейма данных SPI может быть программно изменен. В зависимости от значения бита LSBFIRST в регистре SPI_CR1 данные могут быть

форматированы либо с MSB (most significant bit, старший бит), либо LSB (least significant bit, наименее значимый бит). Каждый кадр данных имеет длину 8 или 16 бит в зависимости от размера данных, запрограммированных с использованием бита DFF в регистре SPI_CR1. Выбранный формат кадра данных применяется для передачи и/или приема. Для sn74hc595 необходимо выбрать размер кадра равным 8. Для данной реализации программы был выбран порядок MSB. Это соответствует установке в регистре SPI_CR1 7 бита (LSBFIRST) равным 0 и бита 11 (DFF: Data frame format) равным 0.

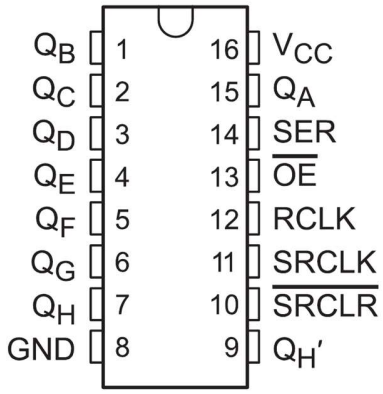

Рис. 50. Выводы sn74hc595 (к рис. 51)

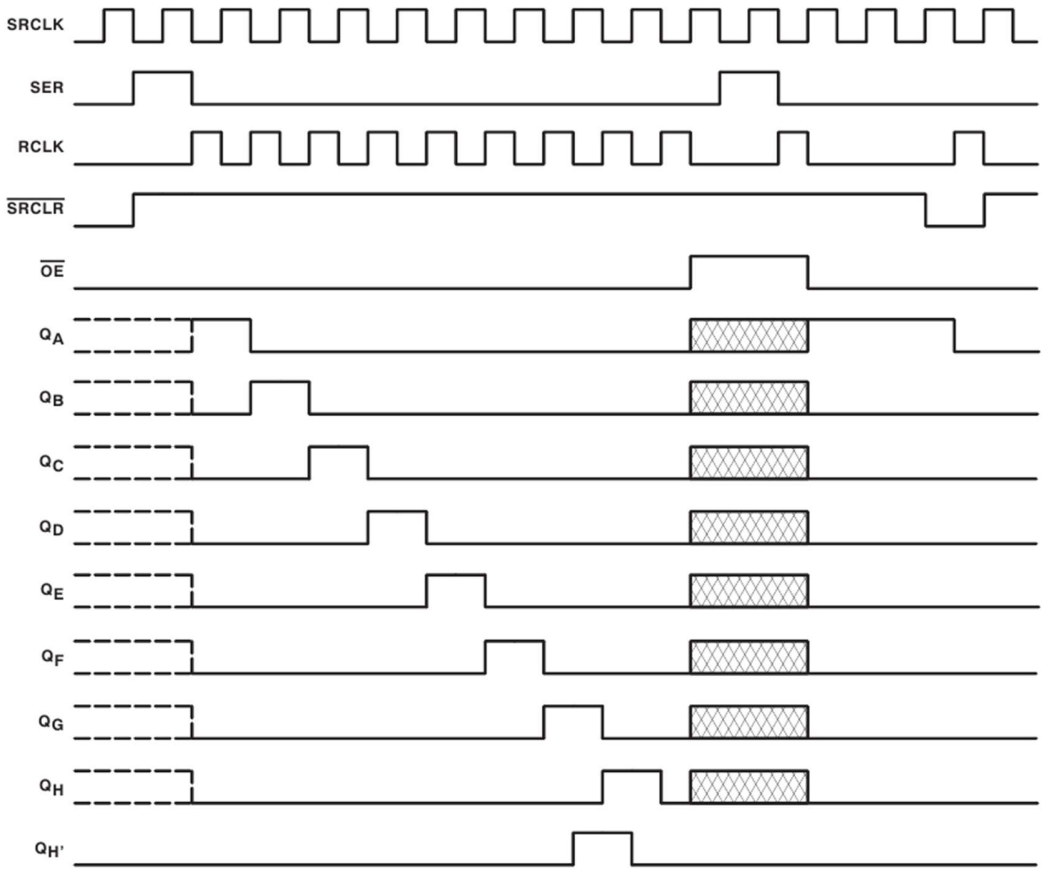

Рис. 51. Временные диаграммы sn74hc595. Где  - 3-е состояние.

Заключение

В ходе работы была разработана система снятия и вычисления стереоизображения и показана возможность построения устройства, позволяющего получать информацию окружающего мира динамически при использовании видеокамер, с изображения которых вычисляется карта глубины. Данное устройство отличается от аналогов независимостью от внешних служб и делает возможным ориентацию в любом пространстве при использовании виброматрицы.

По итогам проведенной работы были исследованы различные технологии и алгоритмы получения стереоизображений при помощи камер и времени распространения сигнала. На основе анализа существующих технологий и алгоритмов, был выбран метод, использующий камеры для снятия изображений. Из рассмотренных алгоритмов обработки изображений для извлечения информации об удаленности предметов был выбран метод, основанный на алгоритме SGBM. Он показал высокие характеристики получаемой карты глубины, которые позволяют построить полностью мобильное устройство, которое будет отличаться простотой ПО и низким энергопотреблением.

Из всех способов построения виброматрицы был выбран метод, основанный на использовании многоканальных ШИМ контроллеров. Данные контроллеры достаточно распространены и имеют низкую стоимость, требуют более простого программного сопровождения. Сравнивая данный метод с методом, основанным на использовании сдвиговых регистров, можно выделить более низкие требования к скорости линии данных, что упрощает и удешевляет необходимое аппаратное обеспечение, а также позволяет с легкостью масштабировать матрицу.

В заключение были продемонстрированы части ПО, отвечающего за работу устройства. На основании всех частей работы возможно построение компактной системы для построения 3D карты удалённости зоны обзора.

Библиографический список

1. A.C. Nau C.P.A.A. Fisher Acquisition of Visual Perception in Blind Adults Using the BrainPort Artificial Vision Device // American Journal of Occupational Therapy, Vol. 69, No. 6901290010p1-8.doi:10.5014/ajot.2015.011809, December 2014.
2. Kaczmarek K.A. The tongue display unit (TDU) for electrotactile spatiotemporal pattern presentation // In: Scientia Iranica / Ed. by Kaczmarek K.A. Elsevier, 2011. pp. 1476-1485.
3. "The vOICe" [Электронный ресурс] URL: https:// www.seeingwithsound.com (дата обращения: 01.03.2021).
4. P Bach-y-Rita C.C.C.F.A.S.B.W.L.S. Vision substitution by tactile image projection // Nature, Vol. 221, 1969. pp. 963-964.
5. M. Parrilla J.J.A.C.F. Digital signal processing techniques for high accuracy ultrasonic range measurements // EEE Trans. Instrum. Meas., vol. 40, Aug. 1991. pp. 759–763.
6. D. Marioli C.N.C.O.D.P.E.S.A.T. Digital time-of-flight measurement for ultrasonic sensors // IEEE Trans. Instrum. Meas., vol. 41, Feb. 1992. pp. 93–97.
7. Regtien C.C.A.P.L. Accurate digital time-of-flight measurement using self-interference // IEEE Trans. Instrum. Meas., vol. 42, Dec. 1993. pp. 990–994.
8. F. Gueuning M.V.C.E.A.P.D. Accurate distance measurement by an autonomous ultrasonic system combining time-offlight and phase-shift methods // in Proc. IMTC, vol. I, , No. Brussels, Belgium, 1996. pp. 399–404.

9. A. Carullo F.F.S.G.U.G.M.P. Ultrasonic distance sensor improvement using a two-level neural network // IEEE Trans. Instrum. Meas., vol. 45, April 1996. pp. 677–682.
10. Carullo Alessio P.M. An ultrasonic sensor for distance measurement in automotive applications // Sensors Journal, IEEE, Vol. 1, No. 10.1109/JSEN.2001.936931, 2001/09/01. pp. 143-147.
11. 13005 E. Guide to the expression of uncertainty in measurement. May 1999.
12. Szeliski D.S.A.R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms // International Journal of Computer Vision, Vol. 1/2/3, No. 47, April-June 2002. pp. 7–42.
13. Tomasi S.B.A.C. Depth discontinuities by pixel-to-pixel stereo // Proceedings of the Sixth IEEE International Conference on Computer Vision Mumbai, India, January 1998. pp. 1073–1080.
14. Hirschmuller H. Stereo Processing by Semiglobal Matching and Mutual Information // In: IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2008. pp. 328 - 341.
15. mathworks. Stereo Disparity using Semi-Global Block Matching [Электронный ресурс] // mathworks: [сайт]. URL: https:// www.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html?searchHighlight=C&s_tid=doc_srchtitle&requestedDomain= (дата обращения: 02.03.2021).
16. H. H. International Conference on Computer Vision and Pattern Recognition // Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. 2005.

17. Spangenberg R. L.T.и.R.R. Computer Analysis of Images and Patterns // Weighted Semi-Global matching and Center-Symmetric Census Transform for Robust Driver Assistance. 2013. pp. 34-41.
18. cv:stereo:StereoBinaryBM Class Reference [Электронный ресурс] // docs.opencv.org: [сайт]. [2021]. URL: https://docs.opencv.org/4.5.2/d7/d8e/classcv_1_1stereo_1_1StereoBinaryBM.html (дата обращения: 02.05.2021).
19. Konolige K. Small Vision Systems: Hardware and Implementation. 333 Ravenswood Avenue, Menlo Park, CA 94025: Artificial Intelligence Center, SRI International,
20. Szeliski R. Structure from motion. In ComputerVision // Texts in Computer Science (Springer), No. 4, Apr 2011. pp. 303–33.
21. R. Zhang P.S.T.J.E.C.A.M.S. Pattern Analysis and Machine Intelligence, IEEE Transactions on Communications // Shape-from-shading: a survey. 1999. Vol. 21(8). pp. 690–706.
22. Hernandez S.S.A.C. IEEE Conference on Computer Vision and Pattern Recognition // Depth from focus with your mobile phone. 2015.
23. Fergus D.E.и.R. Int. Conf.Computer Vision (ICCV) // Predicting depth, surface normals and semantic labels with a common multiscale convolutional architecture. 2015.
24. D. Eigen C.P.и.R.F. In Proc. Conf. Neural Information Processing Systems (NIPS) // Prediction from a single image using a multi-scale deep network. 2014.
25. A. Krizhevsky I.S.и.G.E.H. In Advances in neural information processing systems // В кн.: Imagenet classification with deep convolutional neural networks. 2012. С. 1097–1105.

26. Zisserman K.S.A.A. Very deep convolutional networks for large-scale image recognition // arXiv. 2014. URL: https://arxiv.org/abs/1409.1556 (дата обращения: 21.03.2021).
27. K. He X.Z.S.R.A.J.S. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015. // arxiv.org. 2015. URL: https://arxiv.org/abs/1512.03385 (дата обращения: 04.05.2021).
28. Szegedy S.I.A.C. In Proceedings of The 32nd International Conference on Machine Learning, pages // Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015. pp. 448–456.
29. P. K. Nathan Silberman D.H.A.R.F. 12th European Conference on Computer Vision, ECCV 2012 // Indoor segmentation and support inference from RGBD images. Florence, Italy. 2012. pp. 746-760.
30. A. Dosovitskiy J.T.S.и.T.B. IEEE Conference on Computer Vision and Pattern Recognition // Learning to generate chairs with convolutional neural networks. 2015. pp. 1538–1546.
31. J. Long E.S.и.T.D. IEEE Conference on Computer Vision and Pattern Recognition, pages // Fully convolutional networks for semantic segmentation. 2015. pp. 3431–3440.
32. M. D. Zeiler G.W.T.A.R.F. Computer Vision (ICCV), 2011 IEEE International Conference // Adaptive deconvolutional networks for mid and high level feature learning. 2011. pp. 2018– 2025.
33. Lambert-Lacroix L.Z.A.S. The berhu penalty and the grouped effect. 2012. Документ.
34. V. Belagiannis C.R.G.C.A.N.N. Proceedings of the International Conference on Computer Vision // Robust optimization for deep regression. 2015.

35. Deeper Depth Prediction with Fully Convolutional Residual Networks. Stanford, CA, USA: IEEE, 2016.
36. A. Saxena M.S.и.A.N. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) // Make3d: Learning 3d scene structure from a single still image. 2009.
37. K. Karsch C.L.A.S.B.K. Depth extraction from video using non-parametric sampling. // In: In Proc. Europ. Conf. Computer Vision (ECCV). 2012. pp. 775–788.
38. L. Ladicky J.S.A.M.P. Pulling things out of perspective. // In: Computer Vision and Pattern Recognition (CVPR). 2014. pp. 89– 96.
39. M. Liu M.S.A.X.H. Discretecontinuous depth estimation from a single image // In: onf. Computer Vision and Pattern Recognition (CVPR). 2014. pp. 716–723.
40. B. Li C.S.Y.D.A.V.D.H.A.M.H. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs // В кн.: In Proc. Conf. Computer Vision and Pattern Recognition (CVPR). 2015. С. 1119–1127.
41. F. Liu C.S.A.G.L. Deep convolutional neural fields for depth estimation from a single image // В кн.: Computer Vision and Pattern Recognition (CVPR). 2015. С. 5162–5170.
42. P. Wang X.S.Z.L.S.C.B.P.A.A.L.Y. Towards unified depth and semantic prediction from a single image // In: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. pp. 2800–2809.
43. Todorovic A.R.A.S. Monocular depth estimation using neural regression forest // In: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR (CVPR). 2016.

44. Texas Instruments. SNx4HC595 8-Bit Shift Registers With 3-State Output Registers 1982. URL: https://www.ti.com/lit/ds/symlink/sn74hc595.pdf (дата обращения: 02.03.2021).
45. cypress. Serial Peripheral Interface (SPI) Master 2.20 [Электронный ресурс] [2011]. URL: https://www.cypress.com/file/132086/download (дата обращения: 09.03.2021).
46. Texas Instruments. KeyStone Architecture KeyStone Architecture User Guide // ti.com. 2012. URL: https://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf (дата обращения: 02.05.2021).
47. ST. STM32F103x8, STM32F103xB 2015. URL: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (дата обращения: 03.03.2021).
48. st. RM0008 Reference manual [Электронный ресурс] [2021]. URL: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf (дата обращения: 02.02.2021).
49. texas instruments. ULN2803A Darlington Transistor Arrays [Электронный ресурс] [1997]. URL: https://www.ti.com/lit/gpn/uln2803a (дата обращения: 02.02.2021).
50. NXP. 16-channel, 12-bit PWM Fm+ I2C-bus LED controller Rev. 4 [Электронный ресурс] [2015]. URL: https://www.nxp.com/docs/en/data-sheet/PCA9685.pdf (дата обращения: 06.03.2021).
51. Raspberry Pi hardware [Электронный ресурс] // raspberrypi.org: [сайт]. URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/ (дата обращения: 06.02.2021).

52. Cortex-A72 [Электронный ресурс] // developer.arm.com: [сайт]. URL: https://developer.arm.com/ip-products/processors/cortex-a/cortex-a72 (дата обращения: 23.03.2021).
53. Neon [Электронный ресурс] // developer.arm.com: [сайт]. URL: https://developer.arm.com/architectures/instruction-sets/simd-isas/neon (дата обращения: 09.03.2021).
54. functools — Higher-order functions and operations on callable objects [Электронный ресурс] // docs.python.org: [сайт]. [2021]. URL: https://docs.python.org/3/library/functools.html (дата обращения: 02.05.2021).
55. time — Time access and conversions [Электронный ресурс] // docs.python.org: [сайт]. [2021]. URL: https://docs.python.org/3/library/time.html (дата обращения: 02.05.2021).
56. datetime — Basic date and time types [Электронный ресурс] // docs.python.org: [сайт]. [2021]. URL: https://docs.python.org/3/library/datetime.html (дата обращения: 02.05.2021).
57. 2001 Stereo datasets with ground truth [Электронный ресурс] // 2001 Stereo datasets: [сайт]. [2011]. URL: https://vision.middlebury.edu/stereo/data/scenes2001/ (дата обращения: 04.05.2021).
58. cv:StereoSGBM Class Reference [Электронный ресурс] // https://docs.opencv.org/: [сайт]. [2021]. URL: https://docs.opencv.org/3.4/d2/d85/classcv_1_1StereoSGBM.html (дата обращения: 21.05.2021).
59. Stereo Tuner [Электронный ресурс] // https://github.com/: [сайт]. [2016]. URL: https://github.com/guimeira/stereo-tuner (дата обращения: 07.05.2021).

60. Deeper Depth Prediction with Fully Convolutional Residual Networks [Электронный ресурс] // github: [сайт]. [2017]. URL: https://github.com/iro-cp/FCRN-DepthPrediction (дата обращения: 02.01.2021).
61. Pin Numbering - Raspberry Pi Model A+ [Электронный ресурс] // Pi4J: [сайт]. [2019]. URL: https://pi4j.com/1.2/pins/model-a-plus.html (дата обращения: 02.05.2021).
62. raspberrypi. bcm2711 peripherals // datasheets.raspberrypi.org. 2020. URL: https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf (дата обращения: 02.05.2021).
63. cube-usb-cdc [Электронный ресурс] // github: [сайт]. [2015]. URL: https://github.com/LonelyWolf/stm32/tree/master/cube-usb-cdc (дата обращения: 02.05.2021).