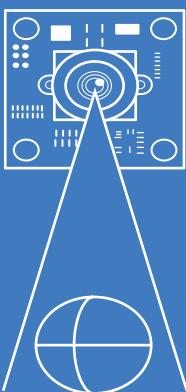
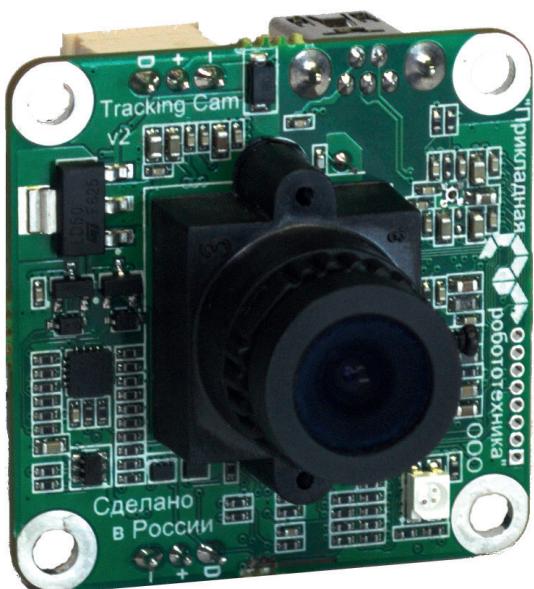


УЧЕБНОЕ ПОСОБИЕ

ТЕХНИЧЕСКОЕ ЗРЕНИЕ РОБОТОВ С
ИСПОЛЬЗОВАНИЕМ
TRACKINGCAM



УДК 004.896

ББК 32.816

к.т.н., доцент МГТУ им. Н. Э. Баумана, Воротников С. А.

к.т.н. Девятериков Е. А.

Панфилов А. О.

Воротников С. А., Девятериков Е. А., Панфилов А. О. Техническое
зрение роботов с использованием TrackingCam / С.А. Воротни-
ков, Е.А. Девятериков, А.О. Панфилов. - Электронная книга, 2017

Данное пособие посвящено изучению основ технического зре-
ния автономных робототехнических систем и мобильных роботов с ис-
пользованием модуля TrackingCam. Рассмотрено понятие «техническое
зрение», применительно к современным робототехническим системам.
Приведен обзор модуля технического зрения TrackingCam и рассмо-
трены процессы настройки и взаимодействия с ним применительно к
учебным робототехническим системам различных брендов. Приведены
примеры программ для распознавания типичных предметов для раз-
личных платформ и в различных средах программирования при реа-
лизации типовых соревновательных задач, рассчитанных на примене-
ние технического зрения. Данное пособие предназначено для изучения
основ робототехники в рамках образовательного процесса дополни-
тельного образования и будет полезно как педагогам, так и учащимся.

© ООО «Прикладная робототехника», 2017

Оглавление

Введение	стр. 4
Глава 1. Что такое «техническое зрение»	стр. 5
Глава 2. Обзор модуля TrackingCam	стр. 12
Глава 3. Программное обеспечение TrackingCam	стр. 14
Глава 4. Обучение и настройка модуля TrackingCam	стр. 24
4.1 Распознавание однотонных областей	стр. 24
4.2 Распознавание разноцветных объектов	стр. 27
Глава 5. Работа модуля с контроллером СМ-530	стр. 29
5.1. Получение данных о распознанных областях	стр. 29
5.2. Получение данных о распознанных объектах	стр. 34
Глава 6. Работа модуля с контроллером OpenCM	стр. 36
Глава 7. Работа модуля TrackingCam с Arduino-совместимым контроллером	стр. 43
Глава 8. Практическая часть	стр. 52
8.1 Следящая платформа	стр. 52
8.2 Следование вдоль сложной линии	стр. 62
Заключение	стр. 73

Введение

Данная книга посвящена вопросам применения технического зрения совместно с робототехническими системами в образовательном процессе, построенном на компонентной базе корейского производителя Robotis иArduino - совместимых систем управления.

В качестве основного компонента рассматривается модуль технического зрения TrackingCam, разработанный и производимый компанией ООО «Прикладная робототехника» в России.

Данный модуль является одним из немногих решений в мире для работы с техническим зрением в образовательном процессе, способным получать и обрабатывать изображение до 30 раз в секунду, что делает его применимым для решения задач, где важным параметром является скорость работы.

Возможность получения результата обработки кадра с помощью различных популярных интерфейсов (Dynamixel, UART, I2C, USB, SPI) позволяет использовать данный модуль совместно с различными аппаратными платформами сторонних производителей, а так же использовать для реализации обмена данными любую популярную среду разработки.

В данном пособии будут рассмотрены как теоретические, так и практические основы технического зрения, применительно к популярному оборудованию и соревновательной деятельности. Данное пособие будет полезно для изучения основ технического зрения в робототехнике как педагогам, так и учащимся.

Глава 1.

Что такое «Техническое зрение»

Современные мобильные роботы функционируют в заранее неопределенной, изменяющейся среде, взаимодействуют с объектами в ней. В процессе движения робот должен оценивать обстановку: измерять собственные координаты, классифицировать окружающие его объекты и определять их положение. Для этого используются различные датчики и системы. Для определения координат робота служат спутниковая навигация, маяки различных принципов действия, колесная одометрия, инерциальные системы, лазерные дальномеры. Окружающие объекты также выделяют с помощью измеряющих различные физические величины датчиков: радиолокационных и ультразвуковых, тактильных, термометрических, химических, а также широкого круга оптических датчиков от фотореле до лазерных дальномеров и телекамер и др.

Особое место среди оптических локационных систем занимают системы технического зрения (СТЗ). Цифровая телекамера в отличие от точечного фотореле или датчика цвета, представляет собой двумерный массив из тысяч миниатюрных оптических датчиков, что позволяет получить намного больше сведений об окружающей среде. Сигналы с этого массива преобразуются в яркость и цвет отдельных дискретных элементов – пикселей, образующих цифровое изображение рабочей сцены. В состав СТЗ входят цифровые телекамеры, вычислительное устройство и комплекс программных средств для анализа изображения, в частности для выделения на нем интересующих объектов.

Рост производительности доступных вычислительных средств в последние десятилетия позволяет извлекать из изображений все больше полезной информации, решать с помощью СТЗ все более сложные задачи в реальном масштабе времени, которые раньше решались с помощью множества других датчиков, в том числе в мобильной робототехнике.

Примером применения СТЗ является автоматизированная перевозка грузов на складах или в производственных цехах с помощью робокаров – самодвижущихся транспортных механизмов (Automated Guided Vehicles – AGV), оснащенных системой технического зрения (СТЗ) и оптическим детектором на базе линейки инфракрасных (ИК) датчиков.

Одной из ключевых задач СТЗ является распознавание обра-

зов – заданных определенными признаками объектов сцены. Частным случаем распознавания, широко используемым на производстве и в сервисной робототехнике, является нахождение в кадре заданных контрастных маркеров или направляющих линий, по которым робокар ориентируется в процессе движения и относительно других объектов. В результате текущее положение робокара определяется на основании данных о положении, размере и ориентации искомого маркера в кадре.

Современные процессоры позволяют решать и более сложные навигационные задачи в заранее неизвестной среде: оценка движения робота по изображениям (Визуальная одометрия) или одновременная локализация и картографирование (SLAM). Тем не менее, производительность вычислителя и ресурсоемкость алгоритмов остаются важнейшими ограничениями при реализации СТЗ, поэтому для решения подобных задач применяются многопроцессорные системы, матричные процессоры, а так же специализированные сигнальные процессоры.

Наиболее распространенной является однопроцессорная (одношинная) структура СТЗ (рис. 1.1), построенная на базе персонального компьютера, выступающего в роли вычислительного модуля.

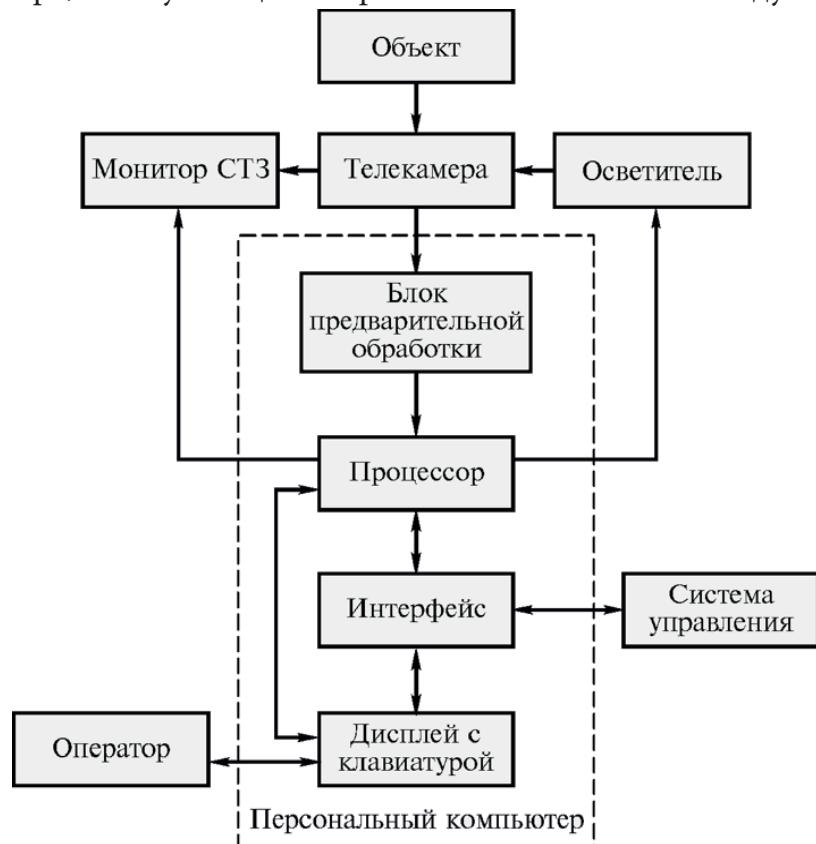


Рис. 1.1. Структура одношинной СТЗ

В целях уменьшения времени на пересыloчные операции из памяти в процессор и обратно потоки информации разделяют, т. е. создают многошинные и многопроцессорные структуры, способные обрабатывать большие потоки видеоданных , поступающих как с одной, так и с нескольких телекамер.

Помимо аппаратной части, в СТЗ не менее важной и сложной частью является соответствующее программное обеспечение, необходимое для ввода и обработки визуальных данных. Современные программные средства в этой области можно разделить на 3 группы:

- инструментальные средства для обращения к аппаратным средствам из приложений(SDK, от англ. software development kit);
- библиотеки с реализацией алгоритмов обработки изображений;
- программные пакеты для ввода и обработки зрительных данных в интерактивном режиме.

Инструментальные средства разработки (SDK) для ввода видеоданных – это комплекс программ, предоставляющих приложениям доступ к устройствам ввода (фреймграбберам) и источникам видеоданных (телекамерам). В настоящее время для обработки изображений в робототехнике получило широкое распространение библиотека OpenCV, состоящая из большого количества алгоритмов, поставляемых с документированным API (от англ. application programming interface – интерфейс программирования приложений или интерфейс прикладного программирования) и примерами приложений на самых популярных языках программирования - C/C++, позволяющих разрабатывать наиболее производительные программы.

Процесс преобразования информации в СТЗ можно представить в виде шести основных этапов:

1. ввод (восприятие) информации, т. е. получение изображения рабочей сцены с помощью датчиков;
2. предварительная обработка изображения с использованием методов подавления шума и удаления искажений;
3. сегментация, т. е. выделение на изображении одного или нескольких представляющих интерес объектов сцены;
4. описание, т. е. определение характерных параметров (цвета, размеров, формы и т. д.) каждого объекта, необходимых для его выделения на сцене;
5. распознавание, или идентификация, объекта, т. е. установление его принадлежности к некоторому классу деталей, например к «болтам»;
6. интерпретация, т. е. получение искомых величин, в частности выяв-

ления принадлежности объекта к группе распознаваемых, например: «на сцене есть несколько гаек».

В соответствии с тем, какие этапы преобразования информации реализуются в конкретной системе, ее можно отнести к СТЗ высокого, среднего или низкого уровня. Так, задачи, решаемые СТЗ низкого уровня, ограничиваются вводом и предварительной обработкой информации, имея на выходе лишь небольшой набор определенных параметров, например - на сцене обнаружено 5 цветовых областей, из которых 2 красные и 3 синие. Для решения задач образовательной и любительской робототехники таких выдаваемых данных, как правило, бывает достаточно.

Говоря о распознавании цветных объектов сцены, необходимо иметь представление цвета в кодировке одной из нескольких цветовых моделей. Цветовая модель — модель описания цветов в виде набора чисел, называемых цветовыми компонентами или цветовыми координатами.

Цветовые модели разделяют на три больших класса:

1. аппаратно-зависимые (описывающие цвет применительно к конкретному устройству цветовоспроизведения — RGB, CMYK);
2. аппаратно-независимые (для однозначного описания информации о цвете — XYZ, Lab);
3. психологические (основывающиеся на особенностях человеческого восприятия — HSB, HSV).

RGB - аддитивная цветовая модель, в которой цвета получаются при смешивании красного, зеленого и синего цветов как лучей источников цвета (черный получается при нулевых значениях составляющих, белый – при максимальных - рис. 1.2 (а)). На рис. 1.2 (б) модель

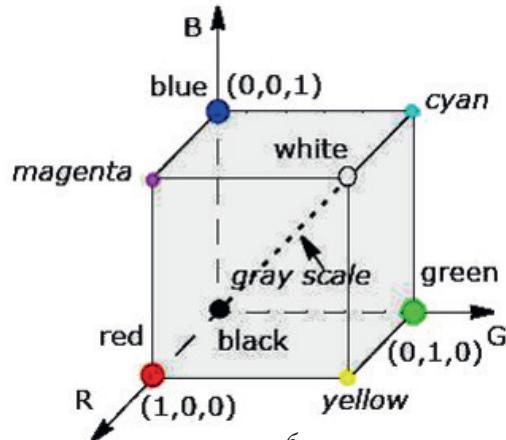
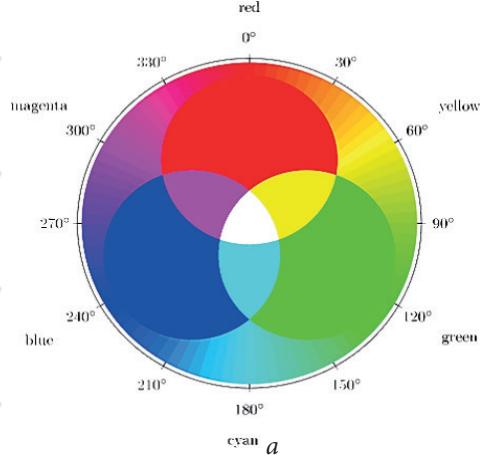


Рис. 1.2. Цветовая модель RGB: а - схема смешения цветов, б - цветовой куб.

представлена в виде цветового куба, где чистые цвета образуют его вершины, а оттенки серого лежат на главной диагонали.

Модель RGB происходит из особенностей физиологического восприятия цвета сетчаткой человеческого глаза и соответствует физическому принципу как получения изображения в цифровых телекамерах, так и отображения информации на устройствах вывода – матрица телекамеры/монитора как правило состоит из фоточувствительных элементов с фильтрами/индикаторами красного, зеленого и синего цвета. Поэтому изображения в формате RGB используются в СТЗ на этапах ввода-вывода, передачи между приложениями, хранения и иногда обработки.

Для описания приемников света применяют модель субтрактивного цветового синтеза CMY (рис. 1.3), основанную на использовании дополнительных цветов голубого, пурпурного и желтого.

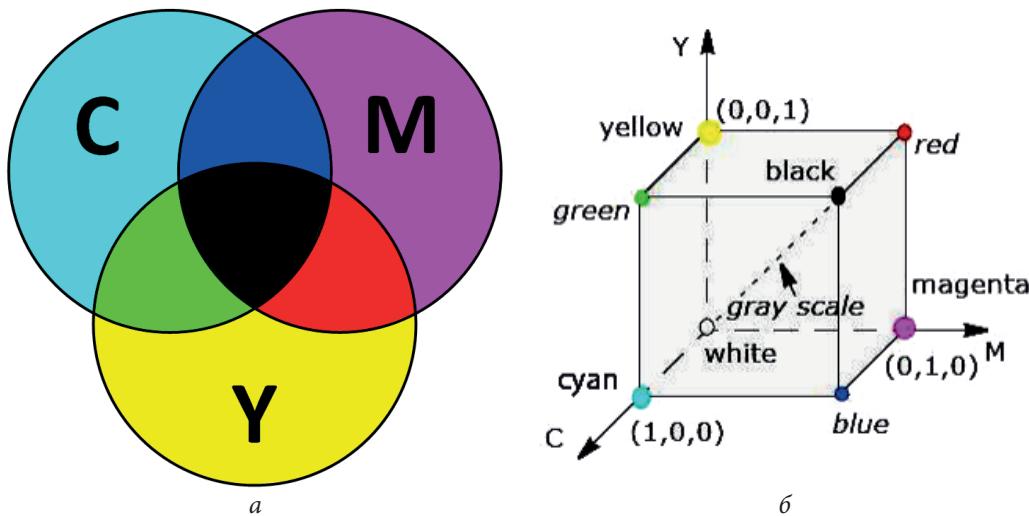


Рис. 1.3. Цветовая модель CMY: а - схема смешения цветов, б - цветовой куб.

Недостатком моделей RGB и CMY является их аппаратная зависимость, что приводит к тому, что цветовое изображение, полученное в CMY (например, при печати на принтере), не совпадает с изображением в RGB, представленным на экране монитора, а также различное восприятие цвета при разной яркости экрана и разной освещенности помещения. Поэтому, в этих моделях сложно выделить такую интуитивно важную характеристику, как «яркость».

Наибольшее количество задач в СТЗ решается при обработке полутонового изображения в оттенках серого, где используется только яркостная информация. Это позволяет выделять контуры объектов, углы, пятна, определять таким образом их форму и положение, что и

требуется в большинстве задач. Цветовая информация в большей степени зависит от условий работы, подвержена шуму, но так же используется. Например, проще пометить известные объекты яркими цветными метками и искать их на изображении, чем анализировать форму этих объектов. По такому принципу цветом кодируются типы объектов и зон в соревновательных задачах для школьного возраста. В таких случаях удобно, когда величины, описывающие цвет объекта, не зависят от яркости – цветовые признаки инвариантны к освещенности.

Такое представление цвета реализуется в перцепционных цветовых моделях, основанных на раздельном представлении информации о цвете и яркости.

Популярная в графических редакторах перцепционная модель HSV (или близкая к ней HSB), напоминает способ, используемый художниками для получения нужных цветов — смешивание белой, черной и серой красок с чистыми красками для получения различных тонов и оттенков. При этом собственно цвет задается с помощью трех приведенных выше независимых показателей — цветового тона H (hue), насыщенности S (saturation) и светлоты V (value) или B (brightness).

В качестве геометрической интерпретации модели HSV используют конус (рис. 1.4, а), полученный как слаженная проекция цветового куба в модели RGB вдоль его главной диагонали (рис. 1.4, б).

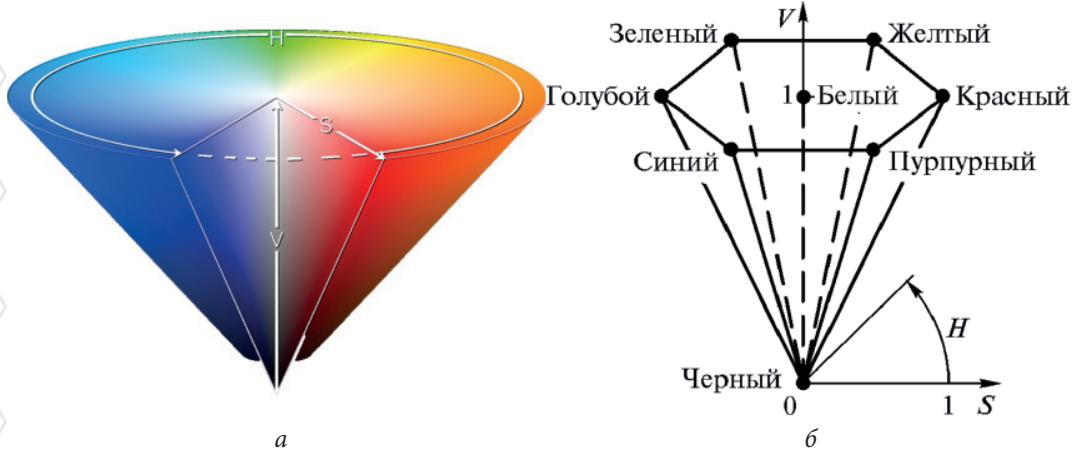


Рис. 1.4. Геометрическая интерпретация цветовой модели HSV: а - цветовой конус, б - проекция куба RGB

Цветовая модель HSV широко используется в интерфейсах настройки СТЗ как интуитивно понятная, а так же при обработке изображений для обеспечения инвариантности цвета объекта к его освещенности. Недостатком является вычислительная сложность преобразований, обусловленная замкнутостью шкалы тона H, а так же пре-

образования исходной информации с матрицы в этот формат.

Распространенной в передаче изображения в СТЗ цветовой моделью, в которой яркость выделена в отдельный канал, являются YCbCr и YPbPr.

В аббревиатуре YCbCr обозначено следующее: Y - компонента яркости, Cb и Cr являются синей и красной цветоразностными компонентами, получаемыми как разность компоненты яркости и соответствующей компоненты цветности. Составляющие Cb и Cr не являются инвариантными к освещенности и не имеют интуитивно понятного смысла, а модель представляет скорее способ кодирования информации, удобный для сжатия и передачи. Тем не менее, СТЗ, принимая изображение в таком формате, может сразу работать непосредственно с яркостной составляющей Y, а при нормировании цветоразностных компонент Cb и Cr по яркости получается инвариантная к освещенности модель с приемлемым цветовым охватом. Такой принцип реализован в модуле TrackingCam.

Наиболее объективную характеристику цвета получают в «аппаратно-независимых» цветовых моделях. Эти модели (к ним относятся модели XYZ и Lab) получили название «математических»; они строятся на основе хроматической диаграммы, разработанной Международной комиссией по освещению CIE (франц. Commission internationale de l'éclairage) и основанной на характеристиках среднестатистического глаза человека. Сама диаграмма представляет собой функцию трех переменных X, Y и Z, описывающих некоторые гипотетические основные цвета. Обычно, модель XYZ используется для измерения способности устройства воспроизвести цвета, с помощью так называемых цветовых охватов – гамутов.

Глава 2.

Обзор модуля TrackingCam

TrackingCam – модуль технического зрения, способный выполнять операции по распознаванию как одноцветных объектов, так и составных объектов, состоящих из нескольких цветовых областей.

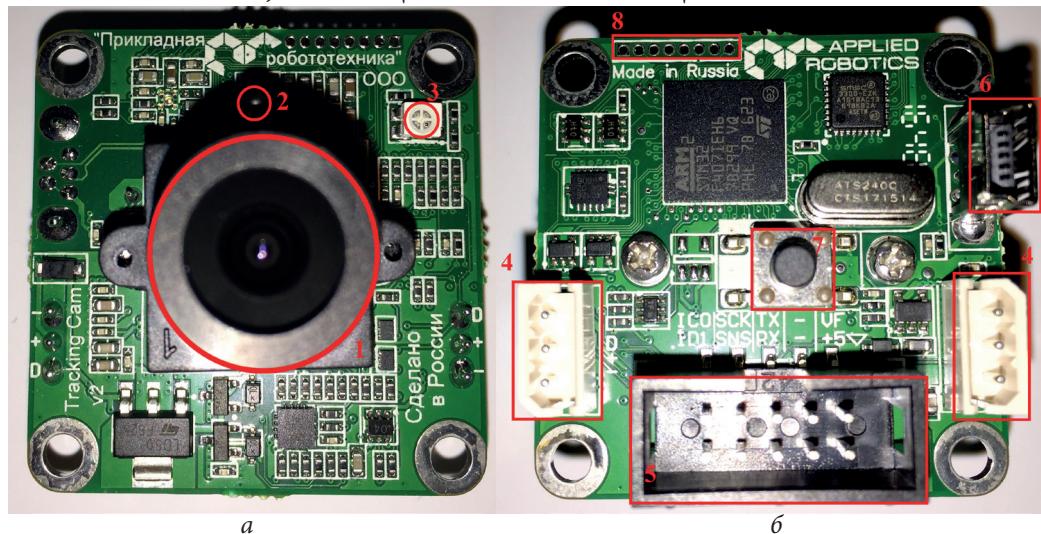


Рис. 2.1. Модуль TrackingCam: а - вид спереди, б - вид сзади

Элементы модуля TrackingCam:

1. Объектив со стандартной резьбой M12
2. Калибровочный винт – используется для фиксации объектива после настройки фокуса
3. Светодиодный индикатор – отображает состояние модуля
4. Разъемы Dynamixel – используются для подключения модуля к конструкторам Robotis как Dynamixel-совместимое устройство
5. Общий разъем подключения к другим устройствам – содержит выводы питания, синхронизации, интерфейсов UART, I2C, SPI.
6. Разъем USB – служит для подключения модуля к ПК
7. Кнопка – используется для перезагрузки телекамеры
8. Отладочные выводы SWD - используются для сервисного обслуживания модуля

Таблица 2.1. Характеристики TrackingCam

Страна-изготовитель	Россия
Матрица	Omni Vision 7725, 1/4"
Разрешение матрицы	640x480 пикселей
Кадровая частота	До 30 кадров в секунду
Процессор	STM32f407 - ARM Cortex M4 168 МГц
Объем ОЗУ	168 Кбайт
Объем флеш-памяти	512 Кбайт
Тип оптики	Стандартная на держатель M12
Углы обзора объективов по горизонтали	От 45 до 75 градусов (в зависимости от используемого объектива)
Углы обзора объективов по вертикали	до 47 градусов (в зависимости от используемого объектива)
Потребляемая мощность	1 Вт
Напряжение питания	5 - 12В
Поддержка интерфейсов	I2C, UART, SPI, Dynamixel, USB
Разъемы	IDC 10 выводов
Поддерживаемые уровни напряжения портов ввода-вывода	3.3В, 5В
Глубина цвета	8 бит
Количество одновременно распознаваемых одинаковых цветовых областей	до 255
Количество запоминаемых образов	до 10 цветных областей и до 5 составных объектов
Количество цветных областей в составном объекте	От 1ой до 3х
Признаки распознавания цветных областей	Яркость, цвет, площадь, округлость, выпуклость, инерция
Признаки составных объектов	Положение и тип основной цветной области, взаимное положение и ориентация цветных областей
Поддержка синхронизации нескольких телекамер между собой	Присутствует, аппаратная с точностью $\pm 1\text{мс}$
Максимальное качество видеопотока на ПК	Разрешение 640x480 пикселей при частоте до 30 кадров в секунду
Поддерживаемые ОС	Windows, Linux
Габариты (ШxВxГ)	38 мм x 38 мм x 32 мм

Глава 3.

Программное обеспечение TrackingCam

Программное обеспечение для работы с модулем на текущий момент предлагается для 2х операционных систем – Windows и Ubuntu и представляет собой интерфейс для задания распознаваемых образов, настройки телекамеры и портов ввода-вывода, захвата изображения и иллюстрации работы алгоритмов (рис. 3.1.).

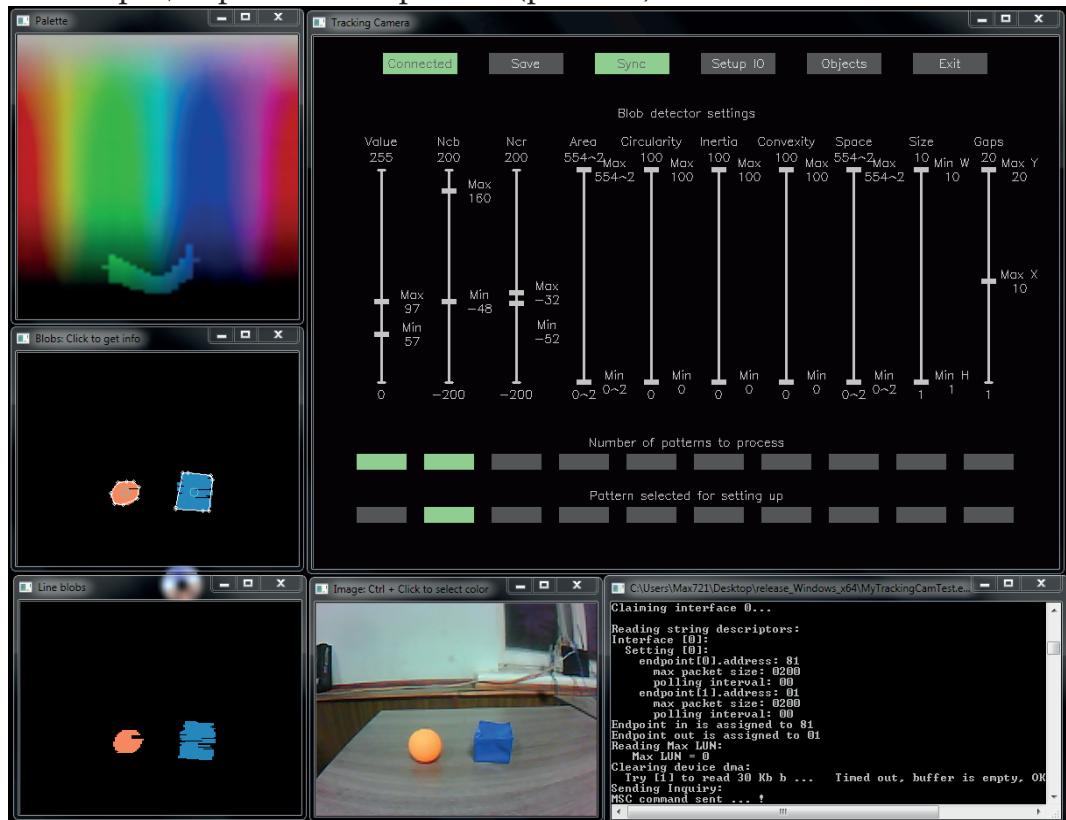


Рис. 3.1. Общий вид интерфейса TrackingCamApp

Рассмотрим каждый из элементов интерфейса подробнее:

Окно с видеопотоком (рис. 3.2.). В него выводится то изображение, которое видит камера модуля. Данное окно не только позволяет наводить модуль на объект, выполняя роль видеоискателя, но, и самое главное, при клике с зажатым ctrl на область модуль сам настраивается с базовыми параметрами на отслеживание этой области. Если кликнуть на этом изображении на оранжевый мяч, то модуль и будет его отслеживать. Сам процесс отслеживания наблюдается на следующих окнах: в окне Line blobs (рис. 3.3.) отображается бинаризация картинки по заданным порогам цвета, т.е. если в зоне видимости камеры будет не-



Рис. 3.2. Окно Image приложения TrackingCamApp



Рис. 3.3. Окно Line blobs приложения TrackingCamApp

сколько различных цветовых областей, соответствующих заданным цветовым настройкам, то в этом окне отобразятся все эти области. Но чаще всего нам необходимо распознавать строго одну область, или область, обладающую определенными признаками – для этого нам необходимо дотянуть настройки системы распознавания и посмотреть на итоговый результат в окне Blobs (рис. 3.4.).

Здесь будут отражены те области, которые соответствуют заданной нами морфологии объекта – размер (площадь) области, ее форма (круг, многоугольник, степень приближения к кругу). Таким образом, мы можем обозначить требуемые нам параметры области (или нескольких областей) и проверить правильно ли они у нас выделяются на фоне остальных распознанных областей. Собственно, данные о тех областях которые мы видим в этом окне и пойдут у нас на вывод по выбранному интерфейсу.

Для настроек модуля существует отдельное окно (рис. 3.5.):

В нем:

Number of patterns to process – количество цветовых шаблонов, которые модуль будет отслеживать. Модуль способен отслеживать одновременно до 10 различных объектов, обладающих различающимися друг от друга характеристиками – цветом, формой, размером.

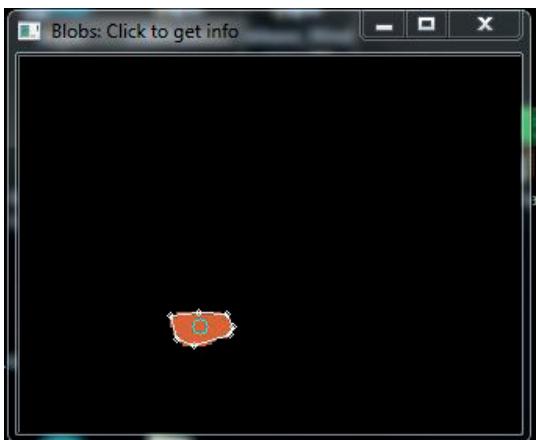


Рис. 3.4. Окно Blobs приложения TrackingCamApp

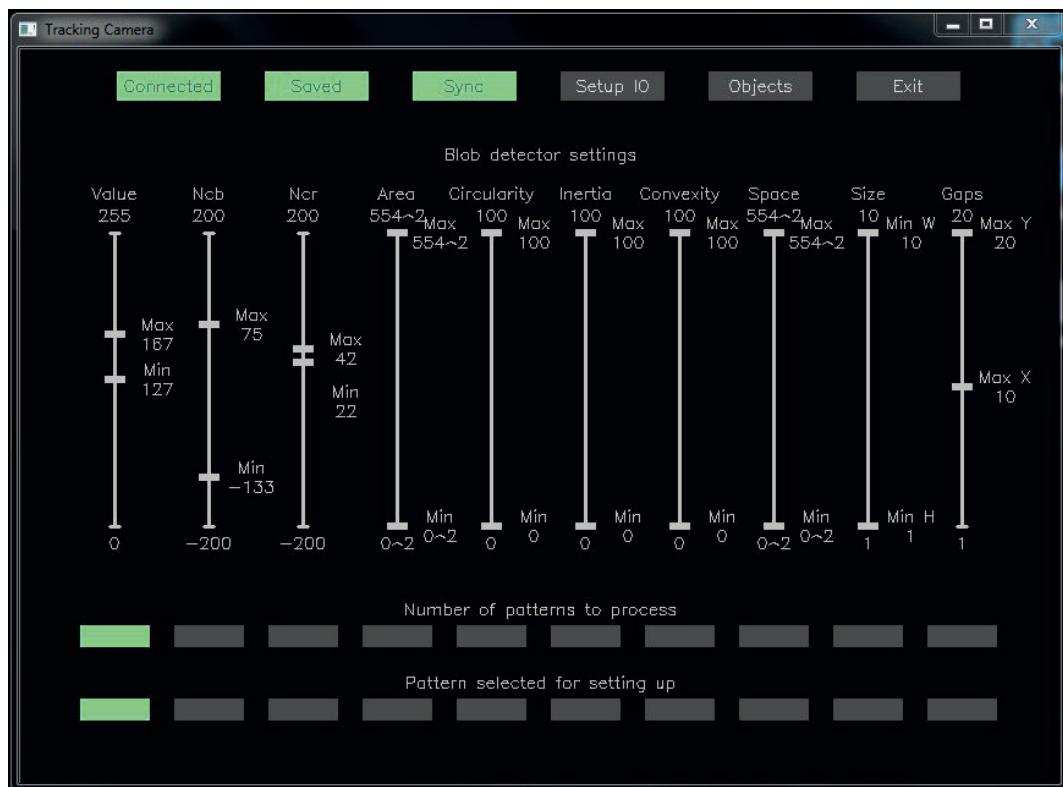


Рис. 3.5. Окно настройки TrackingCam приложения TrackingCamApp

Pattern selected for setting up – выбор шаблона для настройки. Какой шаблон выбран (светится) тот в данный момент доступен для настройки. Параметры для настройки распознавания области следующие:

Настройки цвета для бинаризации:

Value – яркость.

Ncb – синяя цветоразность.

Ncr – красная цветоразность.

Настройки морфологии:

Area – площадь пятна в пикселях. Например, 10^2 – означает площадь 100 пикс.

Circularity – окружность, [0 – угловатая фигура, многоугольник, 100 – круг].

Inertia – вытянутость, [0 – длинная узкая фигура, 100 – круг].

Convexity – выпуклость [0 – внутри габаритного периметра пятна много полостей, 100 – выпуклый многоугольник].

Space – площадь фигуры, ограниченной габаритным многоугольником.

Size – минимальная ширина (W) и высота (H).

Gaps – максимальные разрывы по X и Y между пикселями при объединении в пятна.

Кнопки вверху:

Connect – кнопка подключения к модулю

Need->> - под ней скрывается кнопка записи настроек в модуль, но она становится активной после синхронизации настроек в приложении и в модуле кнопкой Sync (рис. 3.6.).



Рис. 3.6. Окно Sync приложения
TrackingCamApp

При выборе режима синхронизации возможны следующие операции:

- Можно загрузить сохраненные настройки из модуля
- Можно загрузить текущие настройки приложения в модуль
- Загрузить настройки из файла
- Сохранить настройки в файл
- Выход из диалогового окна

В случае выбора режима – загрузить текущие настройки в модуль, в главном окне настроек станет активная кнопка Save вместо кнопки Need.

Кнопка Setup IO - позволяет выполнить настройку интерфейса модуля для обмена данными (рис. 3.7.).



Рис. 3.7. Окно Interface setup приложения TrackingCamApp

Для настройки доступны такие параметры как режим работы UART, скорость обмена данными, адрес (ID) под которым модуль будет видеться в сети устройств Dynamixel, API которым будут выдаваться результаты обработки наружу и напряжение на портах – 3.3 или 5В.

Кнопка Objects открывает окно (рис. 3.8.), с помощью которого можно настраивать композиции областей и формировать из них объекты. В данном случае под объектом будет пониматься совокупность областей, обладающих различными параметрами, но находящимися в строгом соотношении друг относительно друга. Т.е. мы можем обучить наш модуль распознавать не только однотонные пятна, но и области, состоящие из нескольких цветовых пятен (до 3x), которые можно в своих проектах использовать как сложные цветные маркеры.

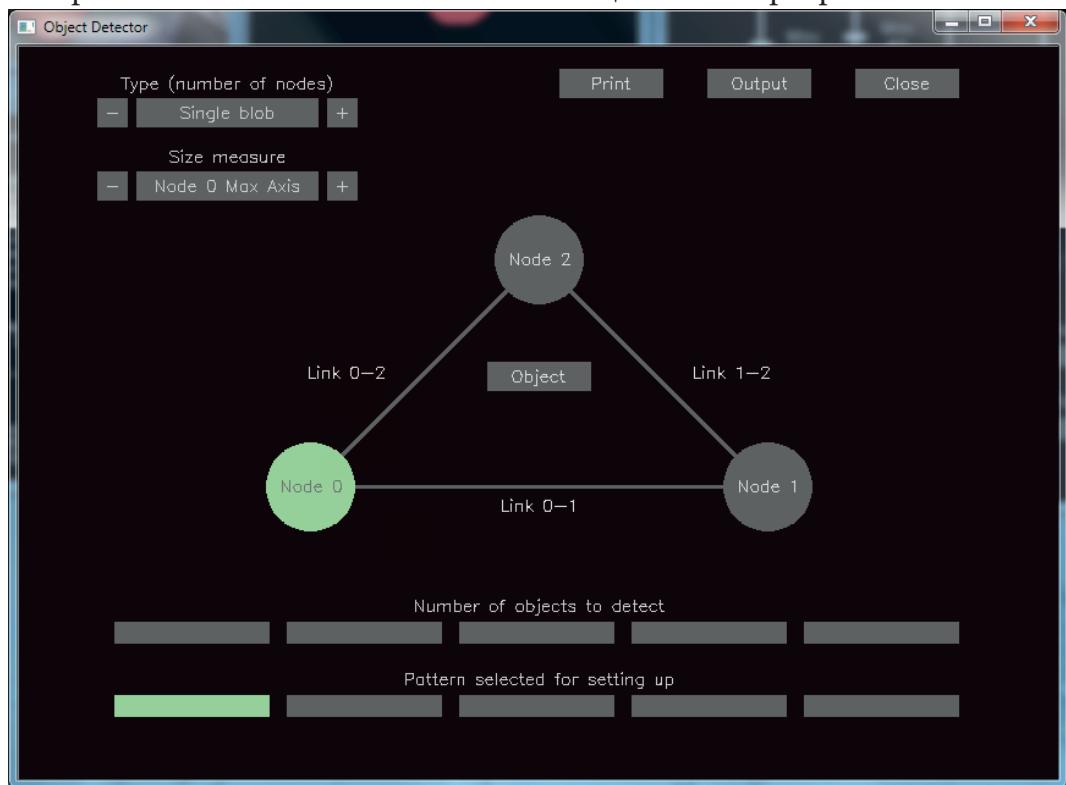


Рис. 3.8. Окно Object Detector приложения TrackingCamApp

В режиме распознавания объектов модуль может распознавать до 5 различных объектов.

Основные элементы окна следующие:

Type (number of nodes) – число узлов, из которых может состоять объект
Size measure – мера размера базового узла Node 0 указывает, что будет пониматься под размером узла при описании структуры. Нужна, чтобы задавать относительный размер связей.

Max axis – большая диагональ габаритного многоугольника пятна.

Width – габаритная ширина пятна.

Height – габаритная высота.

Area – площадь пятна.

Convex area – площадь габаритного многоугольника пятна.

Print – вывести информацию обо всех найденных объектах в консоль.

Output – настройка вывода данных через порт Dynamixel. Sorting criteria

– критерий, по которому объекты сортируются в выходной таблице:

Similarity – по степени соответствия шаблону, наиболее похожие объекты первыми.

Type, similarity – сначала по типу (объекты, соответствующие шаблону 0 первыми), а объекты одинакового типа уже по соответствию.

Output size measure – мера размера всего объекта, которая выводится в таблицу.

Close – выход из окна

Объект может состоять из 3х пятен с обозначенными между собой связями. Настройки параметров пятен осуществляются путем выбора соответствующего узла объекта (рис. 3.9.), а параметры самого объекта задаются во окне Object setup (рис. 3.10.).

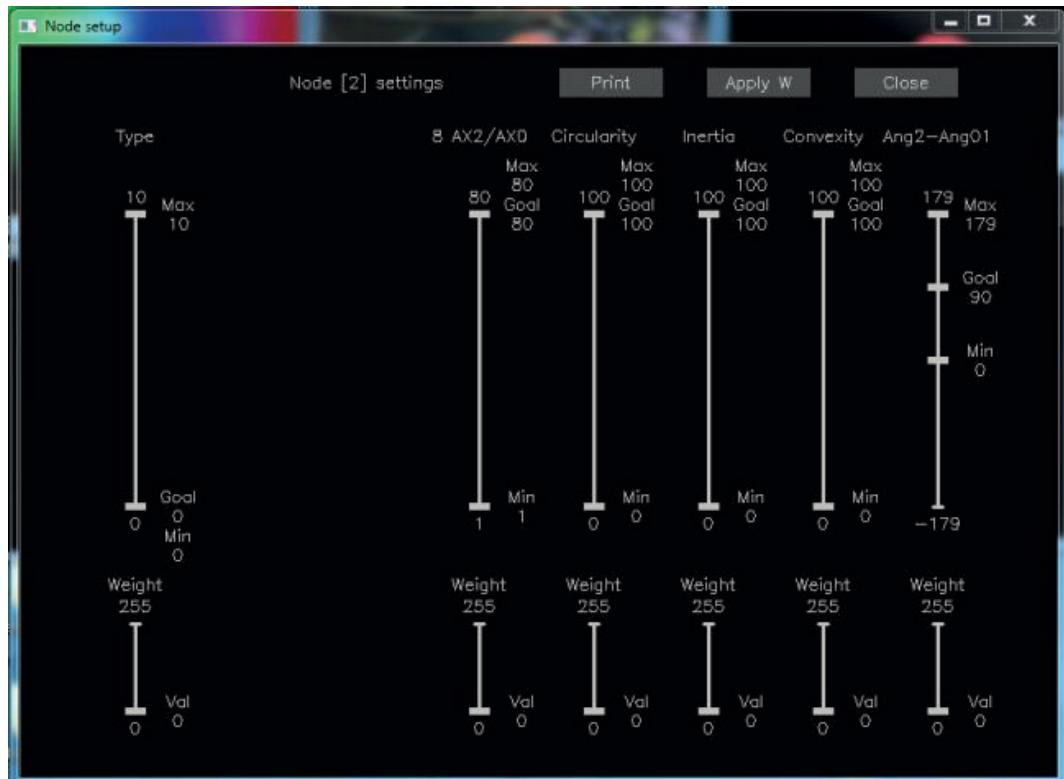


Рис. 3.9. Окно Node setup приложения TrackingCamApp

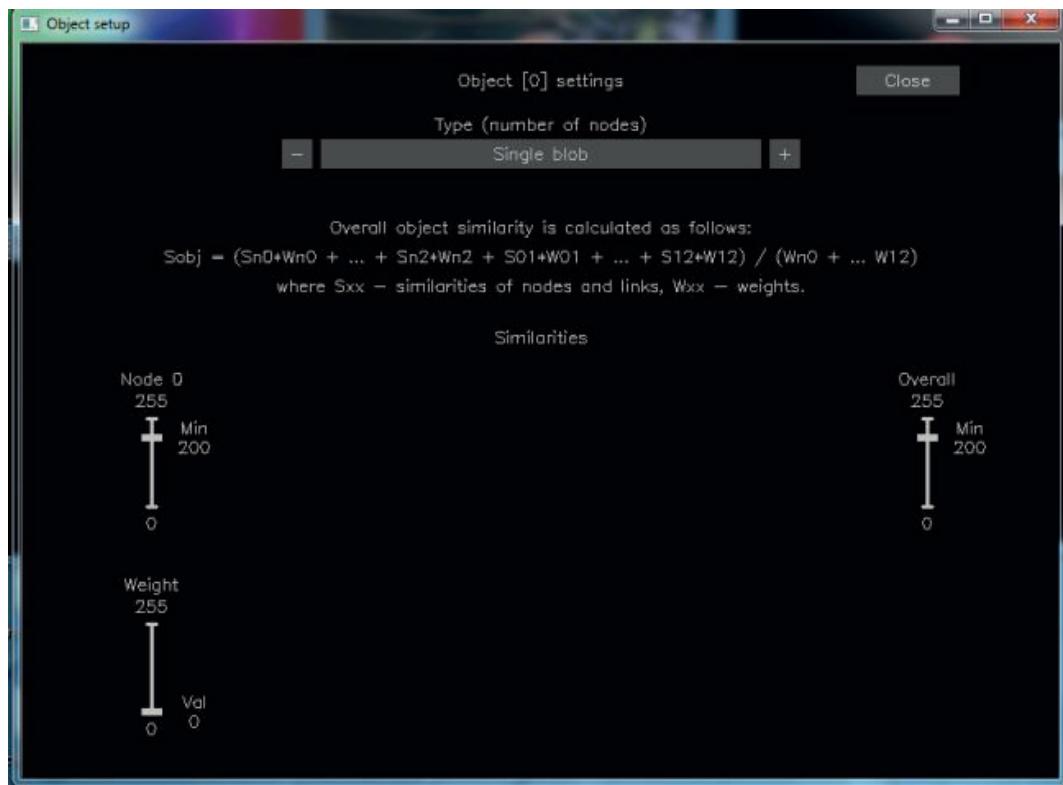


Рис. 3.10. Окно Object setup приложения TrackingCamApp

Все параметры узлов и связей настраиваются с помощью нескольких бегунков:

Goal – целевое значение, при совпадении с которым считается, что степень соответствия объекта шаблону максимальна (100% или 255).

Min и max используются не только как пороговое значение, но и при расчете степени соответствия. Например, если Goal = 90, Max = 200, а реальное значение 150, то степень соответствия параметру:

$$S_{parameter} : (150 - 90) / (200 - 90) = 55\%.$$

Weight – весовой коэффициент при расчете общей степени соответствия узла или связи шаблону. Если вес параметра равен 0, то параметр не учитывается в подсчете соответствия, но все равно проверяются его min и max. Общий коэффициент соответствия узла S_{node} рассчитывается как взвешенное среднее коэффициентов соответствия всех активных параметров:

$$S_{node} = \frac{S_{parameter1}Weight_1 + \dots + S_{parameterN}Weight_N}{255}$$

Параметры базового узла Node 0:

Type – допустимый номер шаблона одноцветной области. Например, [0,

2] будет означать, что узлом могут становиться пятна типов 0, 1 или 2. Сх и Су положение объекта.

Size (мера размера) – абсолютная величина размера базового узла в соответствии с выбранной мерой. Например, Size(Area) – площадь области.

Circularity, Inertia и Convexity аналогичны морфологическим признакам шаблонов одноцветных областей, а здесь задаются, если их необходимо учесть при расчете степени похожести или сортировке. Min и Max дополнительно не проверяются, т. к. отсеяны при обнаружении пятен, но используются как границы при расчете соответствия.

Angle – абсолютный угол поворота узла Node 0. Угол поворота пятна рассчитывается как угол наклона его большей диагонали к горизонту в диапазоне [0°, 180°], т. к. у диагонали нет направления.

Параметры вторичных узлов следующие:

Type, Circularity, Inertia и Convexity – аналогичны базовому узлу.

8 ... (второй бегунок) - задает отношение размера узла к размеру базового узла, а его название меняется в соответствии с заданной мерой размера. Например 8 AX1/AX0

– отношение больших диагоналей 1-го и базового узла. Цифра 8 означает, что модулю задается отношение, умноженное на 8.

Ang1-Ang01 – угол поворота узла Node 1 относительно связи Link01.

Параметры базовой связи Link 01:

L abs – абсолютная длина связи в пикс. в квадрате. Квадрат остается потому, что модуль не извлекает квадратный корень при расчете длины гипотенузы по теореме Пифагора. $(8 \times L/N0size)^2$ – характеризует отношение длины связи к размеру базового узла (в соответствии с заданной мерой). Если мерой узла являются величины с размерностью первой степени: ширина, высота, диагональ и т. д., то они при расчете возводятся в квадрат.

Ang – абсолютный угол поворота связи [0 – слева направо, 360° - справа налево].

Ang to node – относительный угол между связью и большей диагональю базового узла.

Параметры вторичных связей:

L abs, ang – аналогично базовой связи.

$(8 \times L/L01)^2$ – относительная длина связи в сравнении с длиной базо-

вой связи.

Ang from base - относительный угол наклона связи к базовой связи.

Что касается параметров настройки объекта, то они следующие:

Node 0, 1, 2 – минимальная допустимая степень соответствия каждого узла.

Link 01, 02, 12 – минимальная допустимая степень соответствия каждой связи.

Overall – минимальная допустимая общая степень соответствия.

Общая степень соответствия объекта $S_{overall}$ рассчитывается в свою очередь как среднее взвешенное коэффициентов соответствия всех узлов S_{node} и связей S_{link} :

$$S_{overall} = \frac{S_{node0}Weight_0 + \dots + S_{link12}Weight_{12}}{255}$$

Все параметры, отвечающие на настройку цветового восприятия меняют вид цветовой палитры (рис. 3.11.) – светлым выделяется та часть оттенков, которая будет выделяться модулем и затем использоваться для отслеживания области. Таким образом, данное окно используется для удобства настройки, т.к. сразу видно в каких цветовых областях будет выполняться поиск объекта.

В дополнении ко всему вышеуказанному имеется терминал (рис.3.12.), в который выводится:

- Информация об однотонной области при клике на нее в окне Blobs.
- Информация обо всех объектах при нажатии кнопки Print в меню объектов.
- Системная информация.

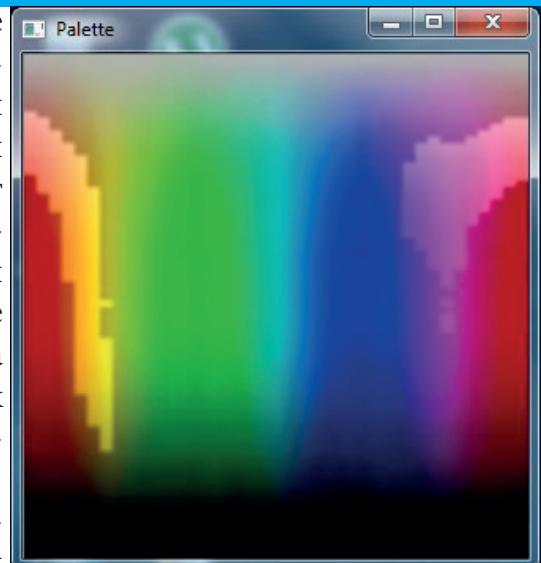
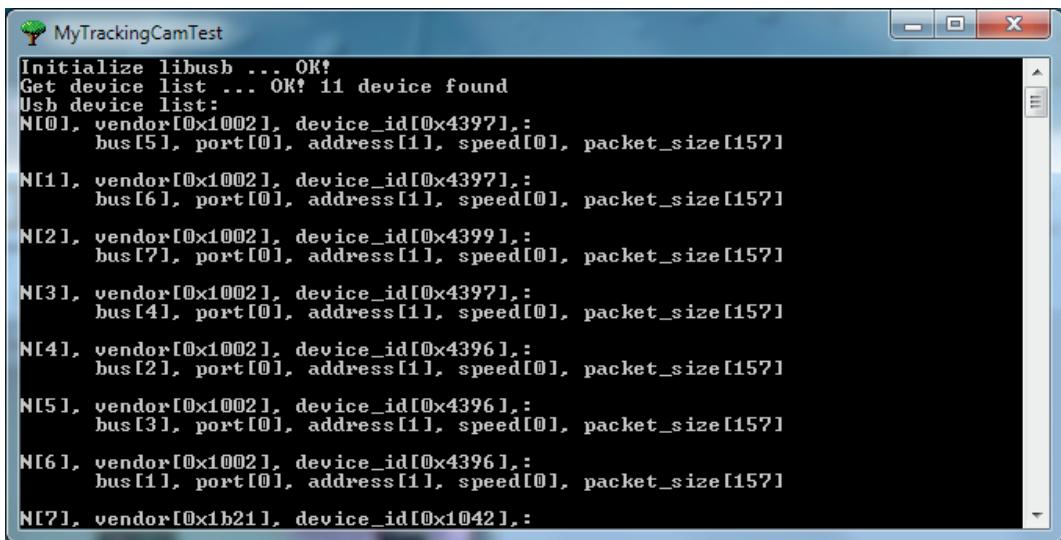


Рис. 3.11. Окно Palette приложения
TrackingCamApp



```
MyTrackingCamTest
Initialize libusb ... OK!
Get device list ... OK! 11 device found
Usb device list:
N[0], vendor[0x1002], device_id[0x4397],:
    bus[5], port[0], address[1], speed[0], packet_size[157]
N[1], vendor[0x1002], device_id[0x4397],:
    bus[6], port[0], address[1], speed[0], packet_size[157]
N[2], vendor[0x1002], device_id[0x4399],:
    bus[7], port[0], address[1], speed[0], packet_size[157]
N[3], vendor[0x1002], device_id[0x4397],:
    bus[4], port[0], address[1], speed[0], packet_size[157]
N[4], vendor[0x1002], device_id[0x4396],:
    bus[2], port[0], address[1], speed[0], packet_size[157]
N[5], vendor[0x1002], device_id[0x4396],:
    bus[3], port[0], address[1], speed[0], packet_size[157]
N[6], vendor[0x1002], device_id[0x4396],:
    bus[1], port[0], address[1], speed[0], packet_size[157]
N[7], vendor[0x1b21], device_id[0x1042],:
```

Рис. 3.12. Окно терминала приложения TrackingCamApp

Данный модуль технического зрения совместим со всеми популярными на сегодняшний день робототехническими контроллерами.

Глава 4.

Настройка модуля TrackingCam

4.1 Распознавание однотонных областей

Настройка камеры для распознавания одноцветной области сводится к заданию ее цветовых и морфологических признаков. Рассмотрим в качестве распознаваемого объекта оранжевый мяч для настольного тенниса, расположим его перед телекамерой таким образом, чтобы он целиком попадал в зону ее видимости (рис. 4.1.). Это необходимо для того, чтобы выбрать все необходимые оттенки цвета, набор которых будет различен, в зависимости от внешнего освещения.

Для выполнения первоначальной настройки необходимо подключить модуль к компьютеру с помощью кабеля USB Mini-B HighSpeed и запустить приложение TrackingCam App. В главном окне нажать кнопку Connect. В окне Image появится изображение с телекамеры. Убедившись, что модуль видит оранжевый шарик целиком, зажав кнопку Ctrl на клавиатуре нажмем левой кнопкой мыши на изображение шарика в окне Image. Модуль отделит шарик от фона и применит к нему параметры распознавания по умолчанию. При этом программное обеспечение автоматически подберет набор цветовых оттенков, которые будут использоваться при распознавании. Выбранные оттенки можно наблю-

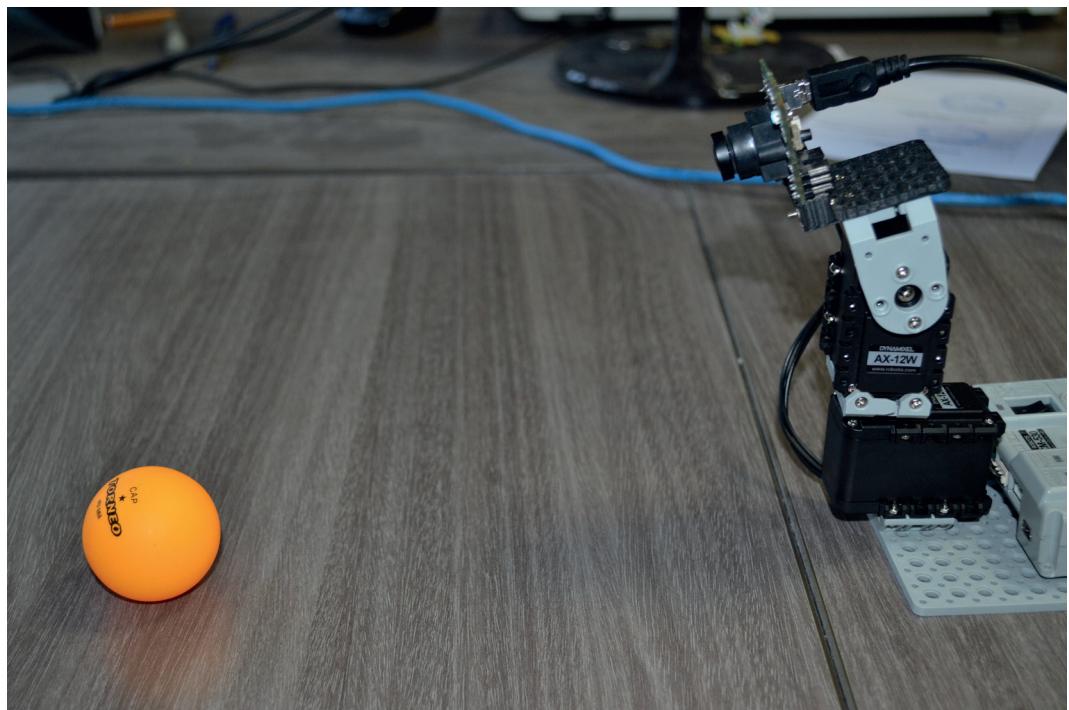


Рис. 4.1. Пример размещения одного однотонного объекта перед камерой

дать в окне палитры (Palette), при необходимости можно расширить область фиксируемых оттенков. Результат поиска выбранных оттенков отобразится в окне иллюстрации бинаризации (Line blobs). Окончательный результат распознавания области с учетом морфологических признаков отобразится в окне Blobs. Если же в окне Blobs не отображается область соответствующая шарику, либо же видно несколько областей, вызываемых прочими внешними объектами или бликами, необходимо отрегулировать настройки детектора (Blob detector settings) в главном окне. После выполнения настройки модуля необходимо выполнить их запись в память модуля, чтобы после выключения питания они сохранялись. Для этого необходимо нажать кнопку Save. После нажатия кнопки Save начнется запись настроек в память модуля, которая займет несколько секунд, после чего кнопка Save сменится на Saved. Если в главном окне вместо кнопки «Save» отображается кнопка «Need-->», необходимо вначале выполнить синхронизацию в меню Sync. Таким образом модуль технического зрения настраивается на распознавание простых однотонных объектов (рис. 4.2.).

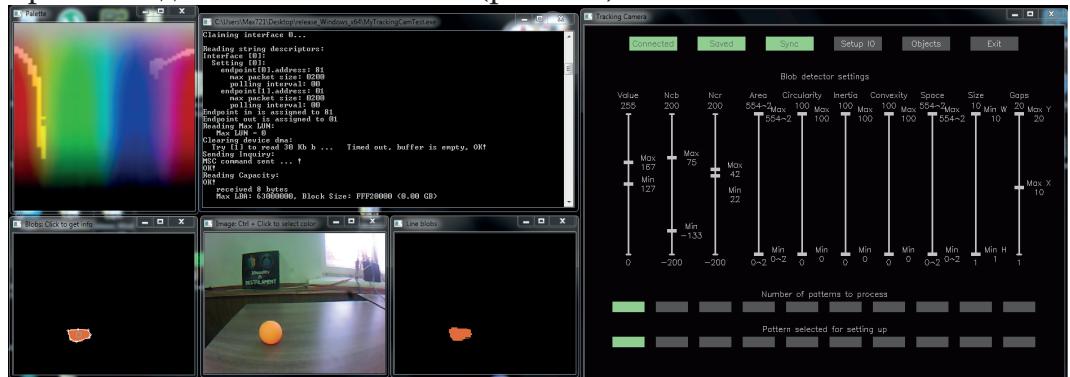


Рис. 4.2. Внешний вид интерфейса TrackingCamApp при настройке на однотонный объект

Модуль TrackingCam способен распознавать до 10 однотонных областей разных оттенков. Для настройки камеры на работу в этом режиме необходимо в главном окне выбрать требуемое число шаблонов переключателем Number of patterns to process - каждый шаблон отвечает за определенный оттенок - и затем по очереди настроить каждый шаблон по аналогии как настраивается модуль на отслеживание и распознавание одной области. Для этого необходимо поочередно выбирать текущий настраиваемый шаблон в переключателем Pattern selected for setting up и последовательно выполнять настройки для каждого требуемого цвета (оттенка). В качестве примера рассмотрим процесс настройки телекамеры на распознавание оранжевого мячика и синего кубика (рис. 4.3.).

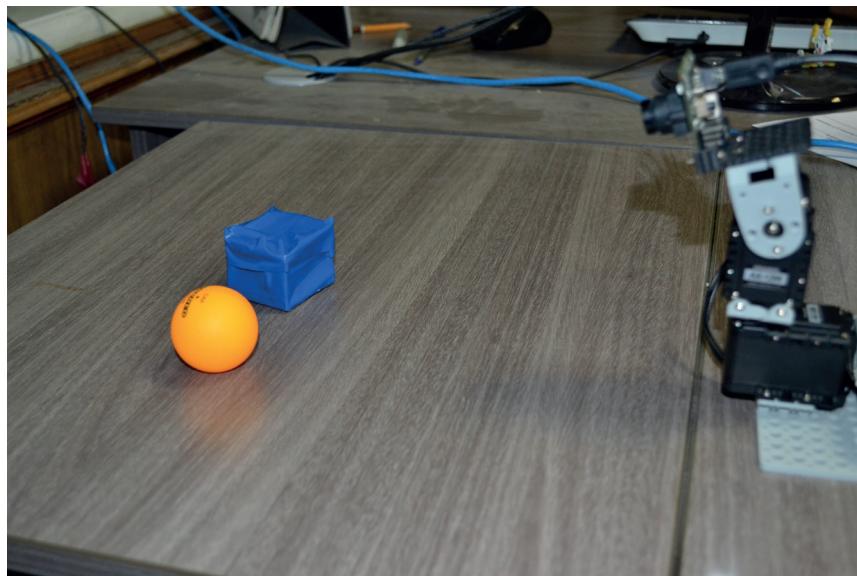


Рис. 4.3. Пример размещения двух однотонных объектов перед камерой

Первый шаблон будет отвечать за распознавание шарика, второй - кубика. После настройки, для получения информации о распознанных областях можно нажать на каждую область в окне Blobs и в консоль будет выведена вся информация о выбранной области. Эта информация будет содержать в себе данные о его типе/шаблоне (type), положении центра (cx, cy), площади в пикселях (area), площади описанного многоугольника (convex area), значения округлости (circularity), вытянутости (inertia), выпуклости (convexity), угол поворота (angle), ширину\высоту (w, h) и длину большей диагонали габаритного многоугольника (max axis). Применительно к ситуации с шариком и кубиком информация может выглядеть следующим образом (рис. 4.4):

```
C:\Users\Max721\Desktop\release_Windows_x64\MyTrackingCamTest...
Blob [1] info:
type [1]
cx [145], cy [201]
area [1350]
convex area [1312]
circularity [66]
inertia [60]
convexity [100]
angle [135]
w [47], h [34]
max axis [50]
Blob [0] info:
type [0]
cx [232], cy [181]
area [580]
convex area [501]
circularity [84]
inertia [62]
convexity [100]
angle [0]
w [33], h [22]
max axis [32]
```

Рис. 4.4. Пример вывода информации о распознанных областях здесь Blob[0] – область соответствующая оранжевому шарику, а Blob[1] – синему квадрату. После выполнения настройки модуля параметры необходимо загрузить в модуль аналогично случаю с распознаванием одной области.

4.2 Распознавание разноцветных объектов

Помимо задач распознавания однотонных областей, зачастую возникает задача отслеживания разноцветных объектов, например, многоцветных меток – маркеров. Для этого модуль TrackingCam настраивается на распознавание составных объектов, которые описываются взаимными размерами, положением, ориентацией и прочими параметрами, входящих в них однотонных областей. При объединении однотонных областей в составные объекты разработчики модуля условно называют их узлами (Node) объекта, а векторы связывающих их параметров связями (Link). Модуль TrackingCam способен отслеживать одновременно до 5 объектов, состоящих максимум из 3-х узлов.

Разберем на примере процесс настройки модуля на распознавание объекта, состоящего из оранжевой и синей областей. Для необходимо аналогично тому, как проводилась настройка на распознавание оранжевой области, настроить его на распознавание еще одной синей области, выбрав 2-й шаблон переключателем Number of patterns to process и 2-й шаблон переключателем Pattern selected for setting up в главном окне. После этих манипуляций в окне Blobs будут отображаться сразу 2-е разных области. Затем необходимо зайти в окно Object Detector (рис. 4.5.), нажав кнопку Object и выбрать из какого количества узлов он будет состоять переключателем «Type (Number of nodes)». В данном

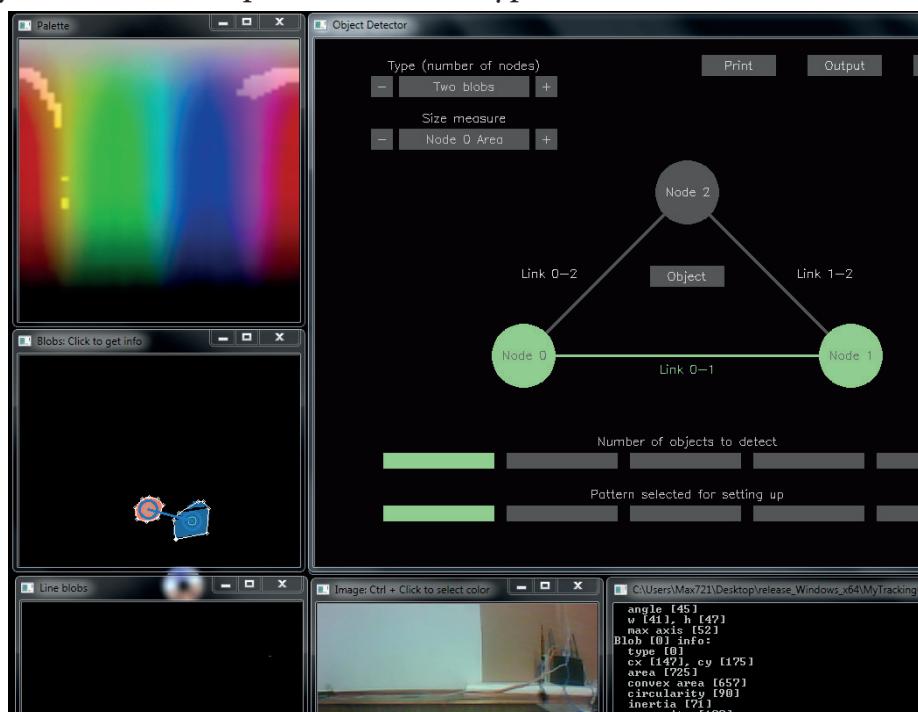
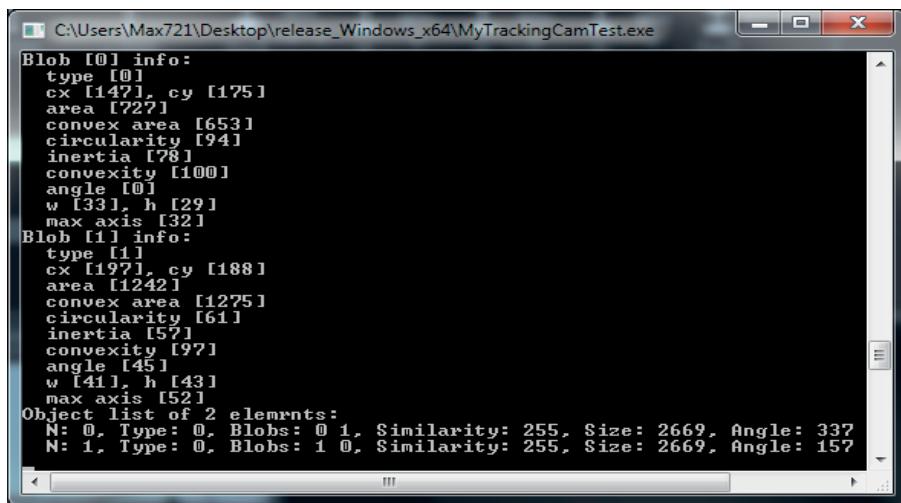


Рис. 4.5. Пример настройки камеры на двухцветный объект

случае объект состоит из 2-х цветных областей (Two blobs). Также необходимо выбрать один шаблон переключателем количества распознаваемых объектов Number of objects to detect. После этого на схеме объекта окажутся подсвеченными 2 узла и образуется связь между ними. При необходимости можно подкрутить их параметры в соответствующих окнах. После настройки параметров искомого объекта необходимо нажать кнопку Close в окне Object Detector и затем в окне Blobs можно будет увидеть оба ранее определенных области, объединенных в один объект. После выполнения настройки модуля параметры необходимо сохранить в модуль аналогичным ранее рассмотренному способом.

Текущую информацию о параметрах как отдельных областей, так и целиком всего объекта можно посмотреть в терминале. Информация о параметрах областей, входящих в состав объекта, получается аналогично случаю распознавания отдельных областей – путем нажатия на интересующую область в окне Blobs. Для вывода сведений обо всех найденных объектах необходимо в окне Object Detector нажать кнопку Print, после чего в консоль будет выведена информация о структуре каждого объекта (из какого количества узлов он состоит) и его узлах: номер шаблона, к которому он относится, вошедшая в него цветная область (Type), порядковые номера цветных областей, степень соответствия распознанного объекта заданному шаблону (Similarity), размер объекта (Size) в единицах заданной меры и угол ориентации (рис. 4.6.).



The screenshot shows a terminal window titled 'C:\Users\Max721\Desktop\release_Windows_x64\MyTrackingCamTest.exe'. The output displays information about two detected blobs:

```
Blob [0] info:  
type [0]  
cx [147], cy [175]  
area [727]  
convex area [653]  
circularity [94]  
inertia [78]  
convexity [100]  
angle [0]  
w [33], h [29]  
max axis [32]  
Blob [1] info:  
type [1]  
cx [197], cy [188]  
area [1242]  
convex area [1275]  
circularity [61]  
inertia [57]  
convexity [97]  
angle [45]  
w [41], h [43]  
max axis [52]  
Object list of 2 elements:  
N: 0, Type: 0, Blobs: 0 1, Similarity: 255, Size: 2669, Angle: 337  
N: 1, Type: 0, Blobs: 1 0, Similarity: 255, Size: 2669, Angle: 157
```

Рис. 4.6. Пример вывода информации об узлах распознанного объекта

Работа модуля в режиме распознавания объектов может быть полезна при решении задач соревновательного направления - распознавания различных цветовых меток, следования по линиям, состоящим из разноцветных областей, классификация окружающих объектов.

Глава 5.

Работа модуля TrackingCam с контроллером СМ-530

5.1 Получение данных о распознанных областях

Модуль TrackingCam является Dynamixel-совместимым устройством, что позволяет подключать его к контроллерам корейского производителя робототехнических комплектующих ROBOTIS без каких-либо дополнительных переходников. Для этого необходимо используя стандартный кабель для подключения сервоприводов Dynamixel соединить модуль TrackingCam и контроллер СМ-530. При этом не имеет значения к какому именно разъему на модуле и к какому именно порту на контроллере будет выполнено подключение, поскольку идентификация Dynamixel-совместимого устройства в системе выполняется по его ID - номеру. По умолчанию модуль TrackingCam имеет ID равный 51. При необходимости этот ID можно изменить при настройке модуля.

При использовании протокола Dynamixel телекамера имеет регистровую модель управления – доступно чтение из памяти телекамеры объемом 256 байт, в которой по адресу 16 (0x10) располагается таблица с данными. Содержание таблицы зависит от режима работы, выбранных алгоритмов обработки изображений и настроек сортировки выходных данных. Если активно только распознавание цветных областей, то данные записываются как показано в таблице 5.1.

Таблица 5.1.
Размещение данных о цветных областях в памяти модуля
при использовании протокола Dynamixel

Адрес	Смещение															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x10	Тип	-	Cx	Cy	S/4	L	R	T	B							
0x20	Тип	-	Cx	Cy	S/4	L	R	T	B							
...																
0xE0	0xFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xF0	0xFF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Тип – номер шаблона, по которому распознана область.

Cx, Cy – положение центра масс области.

S – площадь (делится на 4 для экономии памяти).

L – положение левой границы области по оси X.

R – положение правой границы области по оси X.

T – положение верхней границы области по оси Y.

B – положение нижней границы области по оси Y.

Порядок байт соответствует протоколу Dynamixel – младший байт первый. Информация о распознанных областях размещается подряд в первых ячейках таблицы в порядке обнаружения, пустые ячейки заполняются значением 0xFF (255).

Разберемся как получать данные о распознанных одноцветных областях. Для этого настроим телекамеру на распознавание одной области, например оранжевого мяча, как в рассмотренном ранее примере. После этого отключим модуль от компьютера и подключим его к контроллеру СМ-530. Для написания программы получения данных в данном случае воспользуемся средой разработки RoboPlus Task 2.0.

Создадим новый проект, добавим в него инструкции, обозначающие начало и конец программы и вне основного тела программы создадим функцию TrackingCamParseBlobs - она и будет отвечать за опрос модуля и вывод данных на экран. Следующим шагом внутри функции необходимо указать ID модуля, и начальные адреса памяти с которых начнется его опрос. С помощью инструкции LOAD укажем переменной TrackingCamID константное значение 51. Аналогичным образом укажем адрес, с которого начнется опрос модуля – TrackingCamBlobStorageAddr – его значение 16 – это адрес 1-й строки таблицы данных (первые 16 байт представляют собой системную информацию и для нас интереса не представляют). С помощью инструкции LOAD создадим переменную addr, которой присвоим значение переменной TrackingCamBlobStorageAddr. Переменная addr будет использоваться для записи в нее текущего адреса, чтение которого выполняется, ее значение будет увеличиваться по мере продвижения при чтении таблицы.

Чтобы получить данные о всех найденных областях необходимо перебрать всю результирующую таблицу до первой пустой записи (0xFF). Для этого используем цикл LOOP FOR, который будет выполняться столько раз, сколько будет указано. Поскольку у нашей таблицы 16 строк, то циклу LOOP FOR указываем диапазон итераций – от 0 до 15. Теперь, непосредственно для получения всех данных необходимо перебрать все элементы таблицы построчно сверху вниз. Для этого начнем с того, что с помощью инструкции LOAD прочитаем в переменную BlobType значение, располагающееся по адресу addr Dynamixel-устройства с ID, записанным в переменной TrackingCamID. Такое обращение осуществляется путем выбора пункта Dynamixel Device – Custom и указания в поле ID переменной TrackingCamID и в поле Address переменную addr. После чего с помощью инструкции COMPUTE смещаем адрес на 2 байта, т.к. поле которое читаем состоит из 16 бит. Здесь необ-

ходимо понимать, что в переменную BlobType запишется тип области. Однако, если поле таблицы пустое или произошла какая-либо ошибка, туда могут быть записаны значения 255 (означает пустую запись в таблице) и -1 (ошибка). В случае, если такое случится, то необходимо прервать выполнение цикла LOOP FOR. Для контроля таких ситуаций добавим условный переход IF с 3-мя условиями: если значение переменной BlobType равно 255, -1 или 248 (контроль отсутствия этого значения необходим из-за особенностей протокола Dynamixel). В случае срабатывания одного из этих трех условий выполнение цикла должно прерваться и для этого в условный переход IF добавим инструкцию BREAK LOOP. Проверка на пустые строки таблицы и ошибки реализована.

Продолжим опрашивать модуль. Вне тела условного перехода IF, но внутри цикла LOOP FOR будем создавать конструкции из инструкций LOAD, задача которой считать значение по указанному адресу и присвоить его некой переменной, и COMPUTE, задача которой выполнить смещение адреса. Таким образом выполним присвоение значений следующим переменным: BlobCx – центр области по оси x в пикселях, BlobCy – центр области по оси y в пикселях, BlobArea – площадь области в пикселях, которую еще необходимо домножить на 4 (т.к. при передаче данных значение площади делится на 4, для того чтобы результат поместился в 16 бит), BlobLeft – значение левой границы области в пикселях, BlobRight – значение правой границы области в пикселях, BlobTop – значение верхней границы области в пикселях и BlobBottom – значение нижней границы области в пикселях. Таким образом будут получены значения основных характеристик распознанных областей. Для вывода полученных значений на экран воспользуемся рассмотренными ранее командами Print Screen – для вывода данных без перехода на новую строку и Print Screen with Line – для перехода на новую строку после отображения данных. В результате должна получиться конструкция, изложенная на рисунке 5.1.

Функция для опроса модуля TrackingCam и вывода результатов на экран готова. Теперь необходимо вернуться в тело основной программы и вписать туда вызов функции TrackingCamParseBlobs с помощью инструкции CALL. Еще необходимо понимать, что опрос камеры не происходит мгновенно, так что между вызовами функции TrackingCamParseBlobs необходимо добавить паузу в 0.033 секунды с помощью таймера высокого разрешения (рис. 5.2).

```

1 FUNCTION TrackingCamParseBlobs
2 {
3     TrackingCamID = 51
4     TrackingCamBlobStorageAddr = 16
5     TrackingCamEmptyBlobCode = 255
6     addr = TrackingCamBlobStorageAddr
7     LOOP FOR ( i = 0 ~ 15 )
8     {
9         BlobType = [ ID[TrackingCamID]: ADDR[addr(b)] ]
10        addr = addr + 2
11        IF ( BlobType == 255 || BlobType == -1 || BlobType == 248 )
12        {
13            BREAK LOOP
14        }
15        BlobCx = [ ID[TrackingCamID]: ADDR[addr(w)] ]
16        addr = addr + 2
17        BlobCy = [ ID[TrackingCamID]: ADDR[addr(w)] ]
18        addr = addr + 2
19        BlobArea = [ ID[TrackingCamID]: ADDR[addr(w)] ] * 4
20        addr = addr + 2
21        BlobLeft = [ ID[TrackingCamID]: ADDR[addr(w)] ]
22        addr = addr + 2
23        BlobRight = [ ID[TrackingCamID]: ADDR[addr(w)] ]
24        addr = addr + 2
25        BlobTop = [ ID[TrackingCamID]: ADDR[addr(w)] ]
26        addr = addr + 2
27        BlobBottom = [ ID[TrackingCamID]: ADDR[addr(w)] ]
28        addr = addr + 2
29
30        [ Print Screen ] = BlobType
31        [ Print Screen ] = BlobCx
32        [ Print Screen ] = BlobCy
33        [ Print Screen ] = BlobArea
34        [ Print Screen ] = BlobLeft
35        [ Print Screen ] = BlobRight
36        [ Print Screen ] = BlobTop
37        [ Print Screen with Line ] = BlobBottom
38        [ Print Screen ] = 11111
39        [ Print Screen with Line ] = 11111
40    }
41 }

```

Рис. 5.1. Вид функции *TrackingCamParseBlobs*

```

1 START PROGRAM
2 {
3     ENDLESS LOOP
4     {
5         CALL TrackingCamParseBlobs
6         [ High-resolution Timer ] = 0.033sec
7         WAIT WHILE ( [ High-resolution Timer ] > 0.000sec )
8     }
9 }

```

Рис. 5.2. Вызов функции *TrackingCamParseBlobs*

Программа готова. Если ее загрузить в контроллер робота, открыть окно Debugging и запустить процесс отладки и саму программу на контроллере, то в окне (рис. 5.3.) появятся строки, содержащие информацию о распознанных областях.

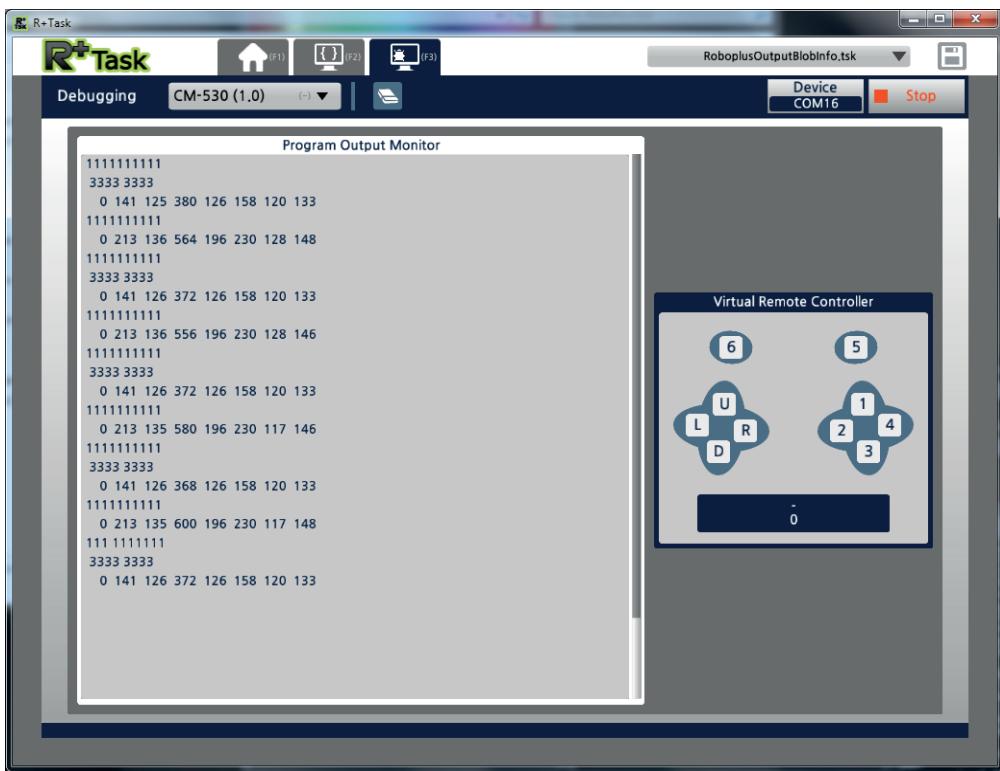


Рис. 5.3. Вывод данных о распознанных областях в окне Debugging

Если в зоне видимости камеры будут находиться 2 одинаковых объекта, то данные будут выдаваться последовательно о двух фиксируемых областях.

Здесь:

0 - BlobType
141 - BlobCx
125 - BlobCy
380 - BlobArea
126 - BlobLeft
158 - BlobRight
120 - BlobTop
133 - BlobBottom
1111111 - разделитель областей
3333 3333 - разделитель запросов

Таким образом в этом примере на экран выводятся данные о двух распознанных областях, относящихся к одному цветовому шаблону. Это видно по параметру BlobType, который у каждой строки одинаков и по различию других параметров - положения области и ее размеров. Таким образом можно получать данные с камеры о распознанных областях и классифицировать их.

5.2 Получение данных о распознанных объектах

Процесс получения данных о распознанных объектах полностью аналогичен процессу получения данных о распознанных одиночных областях за одним исключением – таблица данных заполнена иначе. Если при использовании протокола Dynamixel включено распознавание составных объектов, то данные записываются как показано в таблице 5.2.

Таблица 5.2.

Размещение данных о составных объектах в памяти модуля при использовании протокола Dynamixel

Адрес	Смещение									
	0	1	2	3	4	5	6	7	8	9
0x10	Тип	-	Cx		Cy		S/4		L	
0x20	Тип	-	Cx		Cy		S/4		L	
...										
0xE0	0xFF	-	-	-	-	-	-	-	-	
0xF0	0xFF	-	-	-	-	-	-	-	-	

Тип – номер шаблона, по которому распознан объект.
Cx, Cy – положение начала координат объекта, по умолчанию центр базового узла.
Угол – угол поворота объекта.
Размер – размер объекта в заданной мере (делится на 4 для экономии памяти).

Доступных для чтения данных меньше, потому что сортировка и отбор интересующих объектов настраиваются отдельно и производятся внутри телекамеры.

Обозначим переменные для считывания параметров при работе с объектами следующим образом:

ObjCx – расположение центра объекта по оси x;

ObjCy – расположение центра объекта по оси y;

ObjAngle – угол, на который повернут объект;

ObjSize – размер объекта в выбранной мере

Вывод этих параметров на экран, по аналогии с предыдущим примером, выглядит как на рисунке 5.4.:

Поскольку параметров, доступных для чтения при опросе данных об объектах меньше чем в случае опроса данных об областях, то функция TrackingCamParseBlobs в этом случае будет отличаться только тем, что на экран будет выводить только пять параметров (рис. 5.5.). Во всем остальном функция будет повторять функцию TrackingCamParseBlobs.

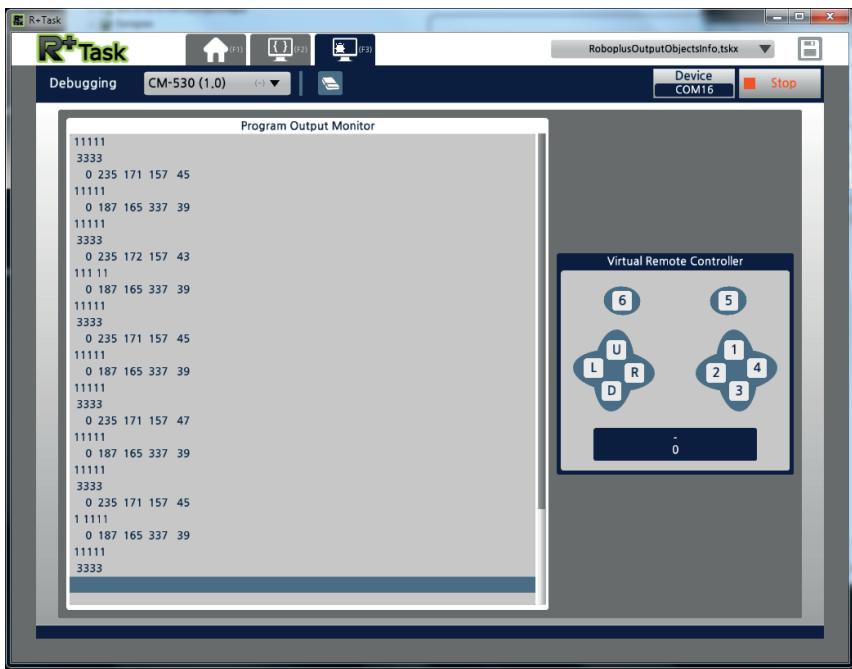


Рис. 5.4. Вывод данных о распознанных объектах в окне Debugging

```

1 FUNCTION TrackingCamParseBlobs
2 {
3     TrackingCamID = 51
4     TrackingCamBlobStorageAddr = 16
5     addr = TrackingCamBlobStorageAddr
6     LOOP FOR ( i = 0 ~ 15 )
7     {
8         ObjType = [ ID[TrackingCamID]: ADDR[addr(b)] ]
9         addr = addr + 2
10        IF ( ObjType == 255 || ObjType == -1 || ObjType == 248 )
11        {
12            BREAK LOOP
13        }
14        ObjCx = [ ID[TrackingCamID]: ADDR[addr(w)] ]
15        addr = addr + 2
16        ObjCy = [ ID[TrackingCamID]: ADDR[addr(w)] ]
17        addr = addr + 2
18        ObjAngle = [ ID[TrackingCamID]: ADDR[addr(w)] ]
19        addr = addr + 2
20        ObjSize = [ ID[TrackingCamID]: ADDR[addr(w)] ]
21        addr = addr + 2
22
23        Print Screen = ObjType
24        Print Screen = ObjCx
25        Print Screen = ObjCy
26        Print Screen = ObjAngle
27        Print Screen with Line = ObjSize
28        Print Screen = 11111
29        Print Screen with Line = 11111
30    }
31
32    Print Screen = 3333
33    Print Screen with Line = 3333
34 }
```

Рис. 5.5. Вид функции *TrackingCamParseBlobs* в случае получения данных о составных объектах

Глава 6.

Работа модуля TrackingCam с контроллером OpenCM

Контроллер OpenCM, является официальным open-source контроллером, поставляемым компанией Robotis для использования совместно со своими комплектующими. По своей структуре и среде разработки контроллер OpenCM очень напоминает Arduino-подобные платы, за исключением того, что его программирование осуществляется из собственной среды разработки Robotis IDE (Robotis OpenCM) (рис. 6.1.). Данная среда очень похожа на Arduino IDE, но все же имеет несколько отличий, в том числе и в синтаксисе команд и функций.

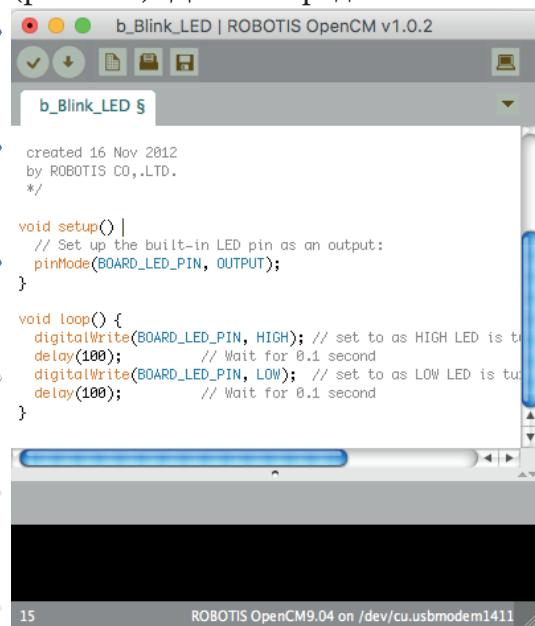


Рис. 6.1. Внешний вид среды разработки
Robotis IDE

Также, в отличии от Arduino IDE Robotis IDE не предоставляет возможности интегрировать сторонние библиотеки стандартными средствами, из-за чего их приходится интегрировать в среду либо путем модернизации файловой структуры каталога Robotis IDE, либо использовать их непосредственно из каталога с разрабатываемой программой.

Несмотря на это неудобство, по умолчанию в среду уже добавлено большое количество библиотек и примеров работы с ними, пред-

назначенных для работы со всеми робототехническими компонентами, предлагаемыми компанией Robotis, а так же с компонентами сторонних производителей, использующих в своих разработках либо элементы от Robotis, либо совместимые протоколы передачи данных.

Таким популярным протоколом является DYNAMIXEL, который используется для обмена данными с сервоприводами Dynamixel, а также с различными сенсорами и другими модулями в большом их разнообразии. Для работы с Dynamixel-совместимыми устройствами в комплекте с Robotis IDE поставляется библиотека с одноименным названием, позволяющая обмениваться данными по протоколу как на низком уровне побайтово, так и прикладными командами, например, вращать

вал сервопривода в определенном направлении с заданной скоростью.

Модуль технического зрения TrackingCam является Dynamixel-совместимым устройством, поэтому его работа с контроллерами OpenCM и CM-530 аналогична, а именно сводится к чтению данных из памяти телекамеры по заданным адресам в соответствующие переменные для дальнейшего использования.

Для использования модуля TrackingCam с контроллером OpenCM рекомендуется выполнять подключение через плату расширения - либо через Expansion Board 485, либо через STEM Board и при работе использовать либо внешний аккумулятор, либо внешний блок питания. Подключение модуля может быть произведено к любому свободному 3x выводному разъему Dynamixel. Варианты подключения TrackingCam приведены на рисунках 6.2 (а) и (б).

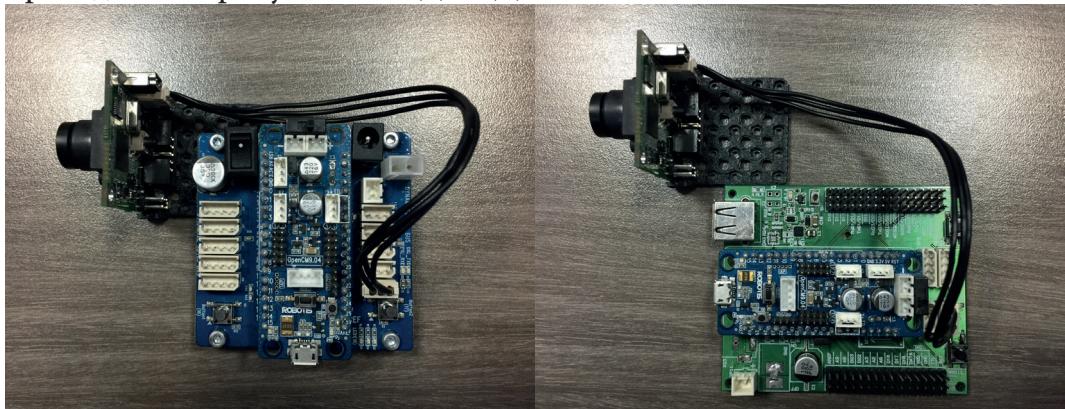


Рис. 6.2. Варианты подключения TrackingCam к OpenCM.

а - через Expansion Board 485, б - через STEM Board

Процесс разработки программы для опроса модуля в обоих случаях подключения будет абсолютно одинаков. Создадим новую программу (скетч) в среде Robotis IDE. Для дальнейшего удобства сразу создадим две структуры: `TrackingCamBlobInfo_t` и `TrackingCamObjInfo_t`. В первой будет переменные для хранения данных о распознанных областях, а в другой - о распознанных объектах. Внутри первой структуры разместим следующие переменные:

- type типа `uint8` для хранения номера шаблона (типа) однотонного объекта
- sx типа `uint16` для хранения значения положения центра объекта по оси X;
- sy типа `uint16` для хранения значения положения центра объекта по оси Y;
- area типа `uint32` для хранения значения площади объекта;

- left типа uint16 для хранения значения левой границы объекта;
 - right типа uint16 для хранения значения правой границы объекта;
 - top типа uint16 для хранения значения верхней границы объекта;
 - bottom типа uint16 для хранения значения нижней границы объекта
- Внутри второй структуры, отвечающей за хранение информации об объектах разместим аналогично переменные :
- type типа uint8 для хранения значения типа объекта
 - cx типа uint16 для хранения значения центра объекта по X
 - cy типа uint16 для хранения значения центра объекта по Y
 - angle типа uint16 для хранения значения угла поворота объекта
 - obj_size типа uint32 для хранения значения размера объекта

Теперь объявим массивы вновь определенных структур:

```
TrackingCamObjInfo_t obj[10];
TrackingCamBlobInfo_t blob[15];
```

В результате часть нашего кода будет выглядеть следующим образом:

```
struct TrackingCamBlobInfo_t {
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint32 area;
    uint16 left;
    uint16 right;
    uint16 top;
    uint16 bottom;
};

struct TrackingCamObjInfo_t {
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint16 angle;
    uint32 obj_size;
};

TrackingCamObjInfo_t obj[10];
TrackingCamBlobInfo_t blob[15];
```

Объявим переменную ID_NUM со значением 51 - ID модуля TrackingCam, переменную addr типа uint16, которая содержит в себе значение адреса памяти с которого необходимо начинать считывание данных и укажем номер интерфейса Dynamixel, к которому подключен модуль. Номер этого интерфейса определяется куда именно подключен

модуль. Если напрямую к плате OpenCM в соответствующий разъем, то его значение 1, если к плате OpenCM и в коммуникационный разъем, то 2, а если к плате Expansion Board или STEM Board, то 3. В нашем случае используется именно последний вариант.

Таким образом 3 последние строчки кода будут выглядеть следующим образом:

```
#define ID_NUM 51  
uint16 addr = 16;  
Dynamixel Dxl(3);
```

Перейдем к заполнению функции `void setup()`. Внутри нее разместим команды инициализации соединения с Dynamixel-совместимым устройством и компьютером, задержку перед началом работы и вывод данных в терминал компьютера о подключенном модулем TrackingCam. Функция в итоге будет выглядеть следующим образом:

```
void setup() {  
SerialUSB.begin(); //инициализация соединения с компьютером по USB  
Dxl.begin(3); //инициализация соединения с Dynamixel-устройством на 3ей скорости  
(1Мбит\с)  
delay(3000); //пауза в 3000 мс  
SerialUSB.println("TrackingCam ID:"); //вывод в терминал на экране компьютера сообщения  
«TrackingCam ID:»  
SerialUSB.println(Dxl.readByte(ID_NUM, 0x00)); //вывод в терминал на экране компьютера  
данных о модуле  
}
```

Для получения данных с модуля используется функция `Dxl.readByte()`, с параметрами `ID_NUM` и адресом считывания.

Для получения данных об однотонных объектах создадим отдельную функцию `TrackingCamGetBlobs()`. В ней будет циклически выполняться следующие операции:

1. считывание данных по указанному адресу;
2. присвоение полученного значения переменной массива, представляющей хранящуюся в телекамере по текущему адресу величину;
- 3) смещение адреса;
- 4) вывод полученного результата в терминал на экране компьютера.

Для реализации этой последовательности поместим внутрь функции переменную `curAddr` типа `uint16`, в которой будет хранится текущий адрес, по которому выполняется считывание. Проинициализируем эту переменную значением переменной `addr` – адресом, по которому в памяти телекамеры хранится таблица текущих данных. После

этого создадим цикл for который будет осуществлять 16 итераций по опросу модуля. Внутри этого цикла сразу разместим вывод на экран значений, полученных в текущей итерации. После этого получим данные о типе объекта и присвоим их соответствующей переменной нашего массива, сместим адрес на 2 байта и выведем на экран полученный тип.

Код для этой операции будет выглядеть следующим образом:

```
uint16 curAddr = addr;
for (uint8 i = 0; i < 16; i++){ SerialUSB.print(i);
SerialUSB.print(" ");
blob[i].type = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].type);
SerialUSB.print(" ");
```

Затем сделаем проверку полученного типа на совпадение со следующими значениями:

- 255 (пустое значение)
- 1 (ошибка)
- 248 (специальный символ)

При совпадении с каким-либо из этих значений необходимо остановить выполнение цикла и выйти из него. Для этого используем оператор break:

```
if (blob[i].type == 255 || blob[i].type == -1 || blob[i].type == 248){
break;
}
```

Если все в порядке и значение поля типа объекта валидное, то далее начинается повторяющийся процесс получения требуемых данных, присвоение полученных значений соответствующим переменным, смещение адреса, вывод полученного значения. Данный процесс будет выполняться до тех пор, пока не закончатся распознанные объектов, либо до тех пор пока не будут получены данные о 16 объектах. Для использования этой функции разместим ее вызов в функции void loop() с паузой в 500 мс.

В результате код программы будет выглядеть следующим образом:

```
struct TrackingCamBlobInfo_t
{
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint32 area;
    uint16 left;
    uint16 right;
    uint16 top;
    uint16 bottom;
};

struct TrackingCamObjInfo_t
{
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint16 angle;
    uint32 obj_size;
};

TrackingCamObjInfo_t obj[10];
TrackingCamBlobInfo_t blob[15];
#define ID_NUM 51
uint16 addr = 16;
Dynamixel Dxl(3);
void setup() {
    SerialUSB.begin();
    Dxl.begin(3);
    delay(3000);
    SerialUSB.println("TrackingCam ID:");
    SerialUSB.println(Dxl.readByte(ID_NUM, 0x00));

}

void loop() {
    TrackingCamGetBlobs();
    delay(500);
}
void TrackingCamGetBlobs (){
    uint16 curAddr = addr;
    for (uint8 i = 0; i < 16; i++){
        SerialUSB.print(i);
        SerialUSB.print(" ");
        blob[i].type = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].type);
        SerialUSB.print(" ");
        if (blob[i].type == 255 || blob[i].type == -1 || blob[i].type == 248){
            break;
        }
    }
}
```

```

blob[i].cx = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].cx);
SerialUSB.print(<< <<);
blob[i].cy = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].cy);
SerialUSB.print(<< <<);
blob[i].area = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].area);
SerialUSB.print(<< <<);
blob[i].left = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].left);
SerialUSB.print(<< <<);
blob[i].right = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].right);
SerialUSB.print(<< <<);
blob[i].top = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2; SerialUSB.print(blob[i].top);
SerialUSB.print(<< <<);
blob[i].bottom = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.println(blob[i].bottom);

```

1	255	0	0	190	119	300	164	255	116	125
1	255	0	0	189	121	268	255	255	116	125
1	255	0	0	192	119	308	158	255	116	125
1	255	0	0	187	120	300	158	255	114	125
1	255	0	0	192	121	236	162	255	116	125
1	255	0	0	190	121	260	166	255	116	124
1	255	0	0	193	120	224	168	255	100	124
1	255	0	0	194	120	260	158	255	116	125
1	255	0	0	193	120	252	168	255	114	124
1	255	0	0	190	121	212	168	255	118	124
1	255	0	0	187	119	396	152	255	100	124
1	255	0	0	192	119	280	160	255	111	124
1	255	0	0	190	118	252	152	255	103	124
1	255	0	0	197	119	252	160	255	118	125
1	255	0	0	192	120	224	162	255	114	124
1	255	0	0	191	119	252	152	255	104	125
1	255	0	0	190	121	248	160	255	112	124
1	255	0	0	193	117	244	170	255	116	124
1	255	0	0	193	121	228	168	255	116	125
1	255	0	0	191	119	288	160	255	114	125
1	255	0	0	192	120	320	160	255	116	125
1	255	0	0	193	119	284	168	255	114	124
1	255	0	0	195	120	264	162	255	114	125
1	255	0	0	194	119	252	162	255	116	124
1	255	0	0	197	121	212	170	255	116	125
1	255	0	0	195	120	220	170	255	114	124
1	255	0	0	194	120	252	170	255	114	124
1	255	0	0	193	120	220	168	255	114	124
1	255	0	0	194	119	276	162	255	114	124
1	255	0	0	195	121	1020	170	255	1	

Рис. 6.3. Вывод данных с OpenCM

Если полученный код запустить и открыть окно терминала, то результат будет аналогичен, представленному на рисунке 6.3.
В нем:

- первое число - последняя итерация, на которой произошла остановка цикла for в результате срабатывания одного из условий.
- второе число - значение типа, полученное на последней итерации (255 - пустое)
- третье число - значение последней итерации на которой была корректно получена информация о распознанной области
- четвертое и далее числа - информация об объекте, представленная в соответствии со структурой TrackingCamBlobInfo.

Таким образом можно получить информацию о распознанной области с помощью контроллера OpenCM.

Процесс получения данных о распознанных сложных объектах будет абсолютно аналогичен процессу получения данных об однотонных объектах, за исключением разницы в количестве получаемых данных, так что позволю себе не приводить пример этой функции.

Глава 7.

Работа модуля TrackingCam с Arduino - совместимым контроллером

Помимо интерфейса Dynamixel для работы с конструкторами Robotis как Dynamixel-совместимое устройство модуль TrackingCam обладает набором стандартных интерфейсов, таких как UART, SPI, I2C, с помощью которых его можно подключать к широкому кругу других вычислительных устройств.

Одним из самых популярных вычислительных модулей в наше время является аппаратная платформа Arduino, во-первых, обладающая широким набором популярных интерфейсов для обмена данными с периферийными устройствами, и во-вторых, доступная для программирования из простой среды разработки - Arduino IDE. Несмотря на существующее в настоящее время обильное разнообразие как оригинальных, так и китайских копий, а так же Arduino-совместимых и Arduino-подобных плат, все эти аппаратные платформы объединяет возможность их программирования из среды Arduino IDE.

Таким образом, процесс использования модуля TrackingCam с любой Arduino-подобной платой практически не отличается от использования модуля с оригинальной Arduino-платой. В качестве примера рассмотрим процесс работы с модулем технического зрения на примере использования оригинальной платы Arduino Mega 2560 и Arduino-подобной платой - программируемым контроллером «ТЕХНОЛАБ». Для обмена данными между модулем и платой остановимся на самом популярном интерфейсе обмена данными - UART. Посколько и Arduino Mega 2560 и программируемый контроллер «ТЕХНОЛАБ» имеют несколько аппаратных интерфейсов UART, то при подключении необходимо понимать к какому именно каналу выполняется подключение модуля TrackingCam. В случае Mega2560 для подключения доступны интерфейсы UART1(Rx1/Tx1), UART2(Rx2/Tx2) и UART3(Rx3/Tx3), а в случае программируемого контроллера «ТЕХНОЛАБ» - UART1(Rx1/Tx1) и UART3(Rx3/Tx3), т.к. UART2 зарезервирован под установленный на борту Bluetooth модуль.

Подключение модуля TrackingCam осуществляется с помощью 4х проводного соединения, где 2 проводника осуществляют питание модуля, а остальные - обмен данными. Со стороны камеры подключение выполняется к выводам 10-пинового разъема как показано на рис. 7.1.

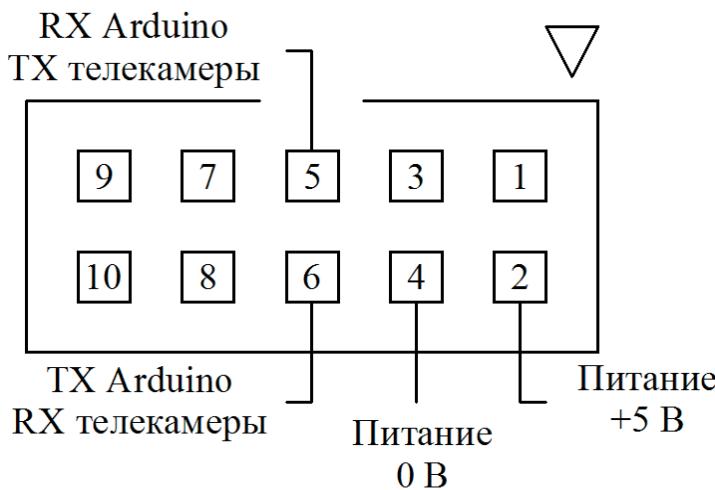


Рис. 7.1. Схема расположения выводов UART интерфейса TrackingCam

Таблица 7.1.

Соединение выводов телекамеры TrackingCam и различных отладочных плат через интерфейс UART

TrackingCam		Обозначение на плате Arduino Mega2560	Обозначение на плате ТЕХНОЛАБ	Назначение
Контакт	Цепь			
2	+5	5V	+5V	Питание +5 В от Arduino
4	-	GND	GND	Питание 0 В от Arduino
5	TX	RX1	UART1 RXD	UART RX Arduino TX телекамеры
6	RX	TX1	UART1 TXD	UART TX Arduino RX телекамеры

ним. Перед тем, как начать опрос модуля необходимо его настроить на распознавание какого-либо предмета, например, оранжевого шарика. Обратите внимание, что не рекомендуется одновременно осуществлять питание модуля и со стороны Arduino и со стороны ПК по USB-кабелю. Перед подключением модуля к ПК для настройки рекомендуется обеспечить вычислительную платформу.

Когда модуль TrackingCam настроен на распознавание какого-либо предмета, его необходимо отключить от компьютера, и подключить

к платам Arduino Mega2560 и ТЕХНОЛАБ телекамера подключается в соответствии с таблицей 7.1.

Несмотря на то, что камера имеет свободный 5ти вольтовый вывод, от него нельзя брать питание для подключения дополнительных внешних устройств, таких как сенсоры или различные периферийные модули.

На стороне Arduino Mega2560 подключение выглядит как показано на рисунке 7.2.

На стороне программируемого контроллера «ТЕХНОЛАБ» подключение выполняется к линии UART1 выглядит как показано на рис 7.3.

После подключения модуля можно приступать к работе с

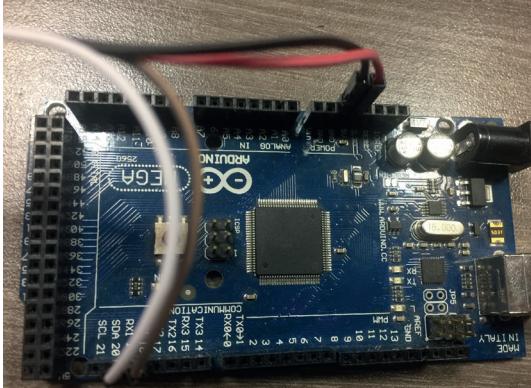


Рис. 7.2. Вариант подключения TrackingCam к Arduino Mega 2560



Рис. 7.3. Вариант подключения TrackingCam к контроллеру ТЕХНОЛАБ

к компьютеру платформу Arduino. После этого на нее сразу пойдет питание и она начнет свою работу в штатном режиме.

Для получения данных о распознанных областях напишем управляющую программу (скетч) в среде Arduino IDE. Несмотря на то, что модуль камеры подключен к плате через интерфейс UART обмен данными ведется с помощью протокола Dynamixel, который необходимо реализовать. Для упрощения работы с модулем TrackingCam с помощью Arduino или Arduino-подобной платы разработана специальная библиотека, задача которой организовывать обмен данными по протоколу Dynamixel и выдавать полученные значения в удобочитаемом виде. вне зависимости от типа платформы Arduino и Arduino-подобной платы существует одна версия библиотеки - для Mega-подобных плат и для UNO-подобных плат, которая называется TrackingCamDxlUart. Для работы с библиотекой рекомендуется использовать Arduino IDE версии не ниже 1.6.5.

Прежде чем начать использование библиотеки, ее необходимо интегрировать в среду разработки. Это делается стандартным способом - в среде Arduino IDE необходимо выбрать пункт меню Скетч - затем Подключить библиотеку, а затем Добавить .ZIP библиотеку... В открывшемся диалоговом окне потребуется указать местонахождение zip-архива с библиотекой и, выбрать ее, нажав кнопку Выбрать. После чего в окне вывода среды появится сообщение об успешном добавлении библиотеки. В результате этих действий библиотека будет интегрирована в среду и появится в списке установленных библиотек, а так же появятся примеры применения этой библиотеки в меню Файл – Примеры - TrackingCamDxlUart (рис. 7.4).

Рассмотрим примеры подробнее. Для использования модуля TrackingCam в качестве примеров предложено рассмотрение двух ос-

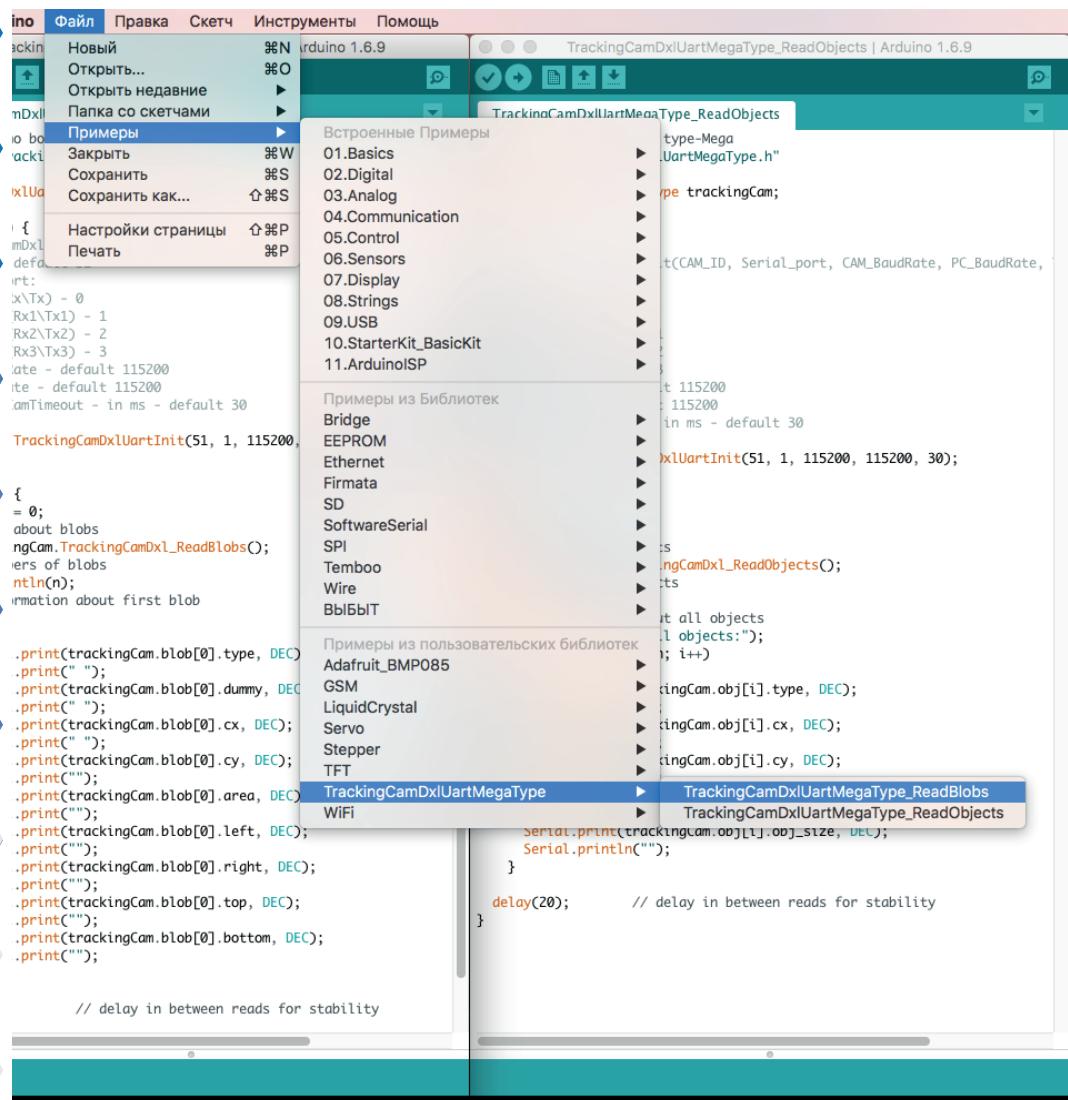


Рис. 7.4. Библиотека TrackingCamDxlUart для Arduino IDE

новных режимов использования модуля: получения информации о распознанных однотонных областях (Blobs), относящихся к одному предмету, либо же получение данных о распознанных объектах - композиции нескольких одноцветных областей. Начнем с примера получения данных о распознанных одноцветных областях.

Поскольку весь процесс обмена данными с камерой по протоколу Dynamixel реализован в библиотеке, то прежде всего в новом скетче необходимо эту библиотеку подключить. Эта команда будет выглядеть следующим образом:

```
#include «TrackingCamDxlUart.h»
```

После этого необходимо проинициализировать модуль технического зрения, путем присвоения ему уникального имени:

```
TrackingCamDxlUart trackingCam;
```

Таким образом, при необходимости, можно подключить несколько модулей к одной плате и обращаться к каждой из них в отдельности по присвоенному уникальному имени. Следующим шагом необходимо выполнить подключение к модулю и ПК для вывода данных на экран. Обратите внимание на этот шаг - если при разработке собственной программы возникнет необходимость выводить какие-либо отладочные данные на экран ПК, то уже не потребуется еще раз устанавливать соединение с помощью Serial.begin(). Подключение модуля TrackingCam выполняется в основной функции скетча - void setup с помощью функции TrackingCamDxlUartInit, которой надо указать следующие параметры:

CAM_ID - ID модуля, как Dynamixel-совместимого устройства - по умолчанию - 51.

Serial_port - номер UART, к которому выполняется подключение модуля- 1, если подключение выполнено к UART1(Rx1/Tx1), 2 - если к UART2(Rx2/Tx2), 3 - если к UART3(Rx3/Tx3). Так же возможно подключение к порту 0 - UART(Rx/Tx), однако, здесь необходимо помнить что этот порт, как правило, зеркалирован с USB выходом Arduino-платы. По умолчанию используется UART1.

CAM_BaudRate - скорость на которой ведется обмен данными между модулем и Arduino-платой. Для настройки этого параметра со стороны камеры необходимо использовать приложение TrackingCam App и в окне Setup указать тип соединения по UART - Full Duplex и скорость обмена данными. По умолчанию рекомендуется скорость 115200.

PC_BaudRate – скорость обмена данными между Arduino -платой и компьютером через USB соединение. Рекомендуется 115200.

TrackingCamTimeout - время ожидания отклика камеры - по умолчанию 30 мс.

Поскольку подключение выполняется строго определенного модуля - по уникальному присвоенному ему имени (trackingCam), то в результате команды подключения будет выглядеть следующим образом:

```
trackingCam.TrackingCamDxlUartInit(51, 1, 115200, 115200, 30);
```

На этом процесс подключения камеры закончен. С ней можно обмени-

ваться данными.

Обмен данными будет реализован в функции void loop, т.к. это бесконечный процесс. Для этого необходимо внутри функции void loop объявить и инициализировать переменную для хранения в ней информации о количестве найденных областей:

```
uint8_t n=0;
```

затем вызовем функцию, отвечающую за получение информации о распознанных областях. В качестве аргумента эта функция возвращает количество распознанных областей, которое мы и запишем в созданную ранее переменную. Выглядеть эта часть кода будет следующим образом:
n = trackingCam.TrackingCamDxl_ReadBlobs();

Для информативности выведем на экран количество найденных областей:

```
Serial.println(n);
```

и в случае если это количество больше нуля выведем информацию о первой найденной области. Для этого используем конструкцию

```
if (n)
{
...
}
```

внутрь которой на место ... поместим последовательность функций Serial.print() в следующих формах:

Получение номера области:

```
Serial.print(trackingCam.blob[0].type, DEC);
```

Пробел между данными:

```
Serial.print(" ");
```

Получение номера шаблона соответствия:

```
Serial.print(trackingCam.blob[0].dummy, DEC);
```

Центр распознанной области по оси X:

```
Serial.print(trackingCam.blob[0].cx, DEC);
```

Центр распознанной области по оси Y:

```
Serial.print(trackingCam.blob[0].cy, DEC);
```

Площадь распознанной области:
Serial.print(trackingCam.blob[0].area, DEC);

Отступ слева:
Serial.print(trackingCam.blob[0].left, DEC);

Отступ справа:
Serial.print(trackingCam.blob[0].right, DEC);

Отступ сверху:
Serial.print(trackingCam.blob[0].top, DEC);

Отступ снизу:
Serial.print(trackingCam.blob[0].bottom, DEC);

Перед завершением цикла void loop рекомендуется добавить паузу в 20 мс с помощью
delay(20);

для организации задержки между опросами модуля и тем самым повышения стабильности выдачи результатов.

На этом программа для обмена данными с модулем TrackingCam и получения данных о распознанных однотонных областях готова.

Следующий пример предназначен для обмена данными с модулем и получения информации о распознанных объектах. Код этого примера отличается от предыдущего только названием функции получения данных - в данном случае используется TrackingCamDxl_ReadObjects() вместо TrackingCamDxl_ReadBlobs() и в массиве с данными распознанных областей - здесь он называется obj[i]. Так же различия заключаются в выводе данных - в данном случае данные выводятся о всех найденных объектах, и поэтому используется цикл for для перебора всех объектов. Что касается данных, которые можно получить об объекте, то здесь они следующие:

Тип объекта:
Serial.print(trackingCam.obj[i].type, DEC);
Центр объекта по оси X:
Serial.print(trackingCam.obj[i].cx, DEC);

Центр объекта по оси Y:
Serial.print(trackingCam.obj[i].cy, DEC);

Угол поворота объекта:

```
Serial.print(trackingCam.obj[i].angle, DEC);
```

Размер объекта:

```
Serial.print(trackingCam.obj[i].obj_size, DEC);
```

Таким образом пример получения данных о распознанных объектах подобен примеру получения данных о распознанных областях. Полный исходный код примеров представлен ниже:

TrackingCamDxlUart_ReadBlobs.ino

```
#include «TrackingCamDxlUart.h»
TrackingCamDxlUart trackingCam; void setup() {
/*TrackingCamDxlUartInit(CAM_ID,      Serial_port,      CAM_BaudRate,      PC_BaudRate,
TrackingCamTimeout);
*   CAM_ID - default 51
*   Serial_port:
*   Serial (Rx\Tx) - 0
*   Serial1 (Rx1\Tx1) - 1
*   Serial2 (Rx2\Tx2) - 2
*   Serial3 (Rx3\Tx3) - 3
*   CAM_BaudRate - default 115200
*   PC_BaudRate - default 115200
*   TrackingCamTimeout - in ms - default 30
*/
trackingCam.TrackingCamDxlUartInit(51, 1, 115200, 115200, 30);
}
void loop() { uint8_t n = 0;
//read data about blobs
n = trackingCam.TrackingCamDxl_ReadBlobs();
//print numbers of blobs
Serial.println(n);
//print information about first blob
if(n)
{
Serial.print(trackingCam.blob[0].type, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].dummy, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].cx, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].cy, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].area, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].left, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].right, DEC);
Serial.print(« »);
```

```
Serial.print(trackingCam.blob[0].top, DEC);
Serial.print(« »);
Serial.print(trackingCam.blob[0].bottom, DEC);
Serial.print(« »);
}

delay(20);      // delay in between reads for stability
}

TrackingCamDxlUart_ReadObjects.ino
#include «TrackingCamDxlUart.h»
TrackingCamDxlUart trackingCam; void setup() {
/*
 * TrackingCamDxlUartInit(CAM_ID, Serial_port, CAM_BaudRate, PC_BaudRate,
TrackingCamTimeout);
*      CAM_ID - default 51
*      Serial_port:
*      Serial (Rx\Tx) - 0
*      Serial1 (Rx1\Tx1) - 1
*      Serial2 (Rx2\Tx2) - 2
*      Serial3 (Rx3\Tx3) - 3
*      CAM_BaudRate - default 115200
*      PC_BaudRate - default 115200
*      TrackingCamTimeout - in ms - default 30
*/
trackingCam.TrackingCamDxlUartInit(51, 1, 115200, 115200, 30);
}
void loop() { uint8_t n = 0;
//read data about objects
n = trackingCam.TrackingCamDxl_ReadObjects();
//print numbers of objects
Serial.println(n);
//print information about all objects
Serial.println(«\nAll objects:»);
for(int i = 0; i < n; i++)
{
Serial.print(trackingCam.obj[i].type, DEC);
Serial.print(« »);
Serial.print(trackingCam.obj[i].cx, DEC);
Serial.print(« »);
Serial.print(trackingCam.obj[i].cy, DEC);
Serial.print(« »);
Serial.print(trackingCam.obj[i].angle, DEC);
Serial.print(« »);
Serial.print(trackingCam.obj[i].obj_size, DEC);
Serial.println(« »);
}

delay(20);      // delay in between reads for stability
}
```

Глава 8.

Практическая часть

8.1 Следящая платформа

Следящие, они же наклонно-поворотные (pan and tilt) платформы широко используются для установки на них телекамер и создания следящих систем. Такая платформа представляет собой устройство из двух сервоприводов, системы управления и элементов крепежа. В нашем случае вместо телекамеры будет использоваться модуль TrackingCam, в качестве системы управления – робототехнический контроллер СМ-530, в качестве сервоприводов – DYNAMIXEL AX-12W, а само крепление модуля к сервоприводам напечатано на 3D принтере.

Цель практического занятия: освоение применения технического зрения в качестве датчика следящих систем управления; повышение навыков работы с микроконтроллерными модульными вычислительными устройствами и датчиками, навыков программирования в области обмена данными между ними; получение частных навыков: освоение модуля TrackingCam и повышение навыков работы с робототехническими конструкторами Robotis.

Задача: из конструктивных элементов от Robotis, сервоприводов Dynamixel и модуля TrackingCam собрать и запрограммировать платформу, направляющую телекамеру на заданный предмет.

Последовательность выполнения задания:

1. Собрать платформу на основе контроллера СМ-530, приводов Dynamixel и модуля технического зрения TrackingCam .
2. Настроить контроллер СМ-530 и приводы Dynamixel в утилите RoboPlus Manager.
3. Настроить телекамеру TrackingCam на распознавание мяча в программе TrackingCam App.
4. Написать программу системы управления для контроллера СМ-530 в среде R+Task 2.0.
5. Проверить работу следящей платформы.

Сборка платформы.

Процесс сборки такой следящей платформы (рис. 8.1 (а), (б)) достаточно прост и показан далее (рис. 8.2.):

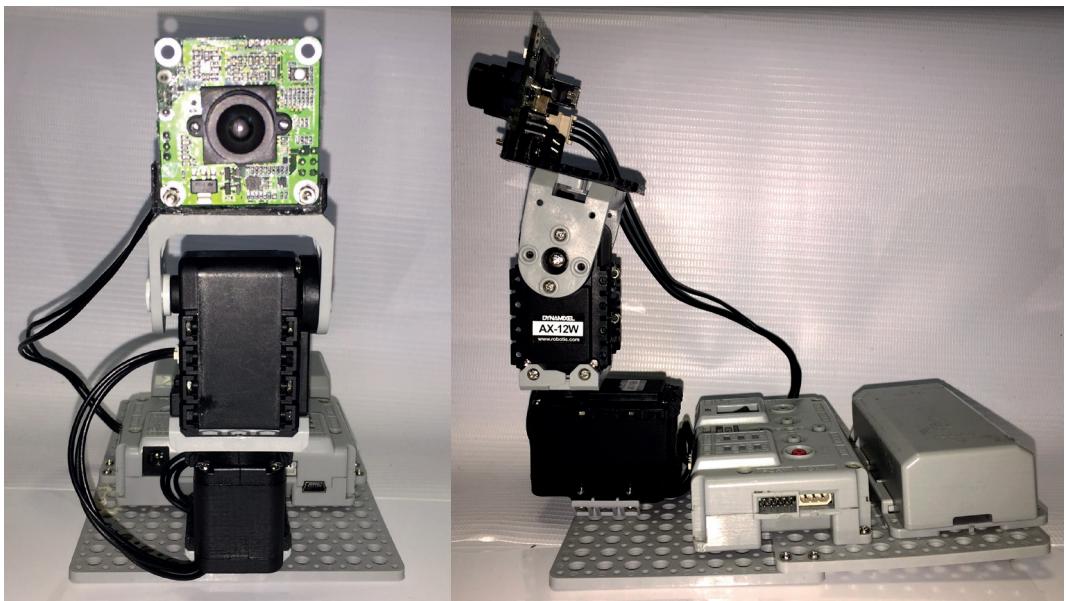


Рис. 8.1. Внешний вид следящей платформы. а - вид спереди, б - вид сбоку.

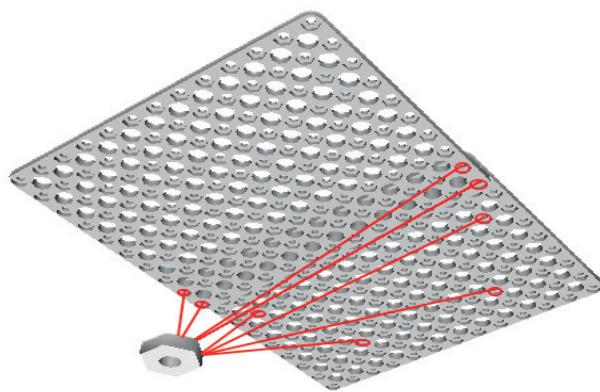


Рис. 8.2. (а) Сборка следящей платформы. Шаг 1.

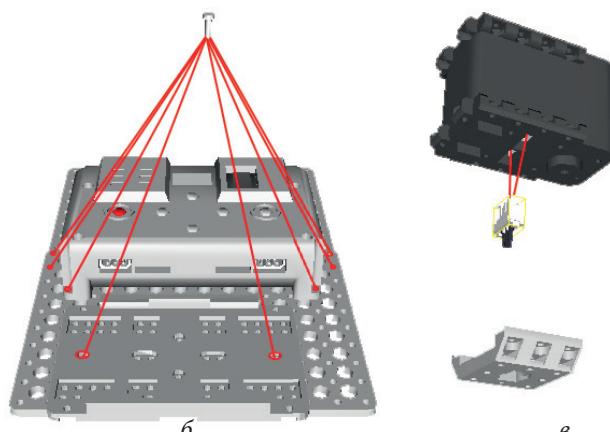


Рис. 8.2. Сборка следящей платформы. б - шаг 2, в - шаг 3.

Шаг 1.

Соединяем 2 пластины 9x12 как показано на рисунке 8.2 (а).

Шаг 2.

Устанавливаем на пластины контроллер СМ-530 и держатель аккумулятора (рис. 8.2 (б)).

Шаг 3.

Выполняем подключение 2x Dynamixel-кабелей к первому сервоприводу (рис. 8.2 (в)).

Шаг 4. Устанавливаем и закрепляем на сервоприводе крепежную пластину (рис. 8.2. (г)).

Шаг 5. Закрепляем собранную конструкцию на платформе с контроллером и аккумулятором (рис. 8.2. (д))

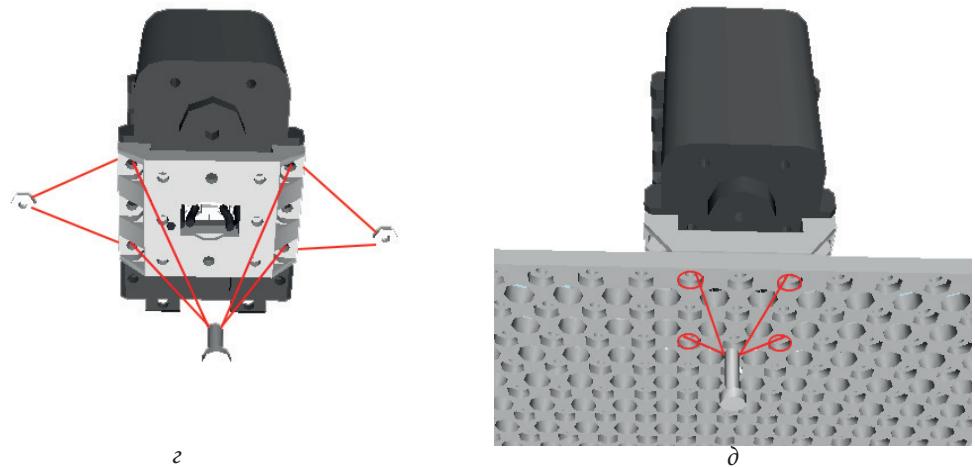


Рис. 8.2. Сборка следящей платформы. г - шаг 4, д - шаг 5.

Шаг 6. Устанавливаем на первый сервопривод пластину для крепления второго сервопривода (рис. 8.2. (е)).

Шаг 7. Устанавливаем второй сервопривод (рис. 8.2. (ж), (з)).

Шаг 8. Устанавливаем на второй сервопривод поворотную платформу для крепления модуля технического зрения (рис. 8.2. (и))

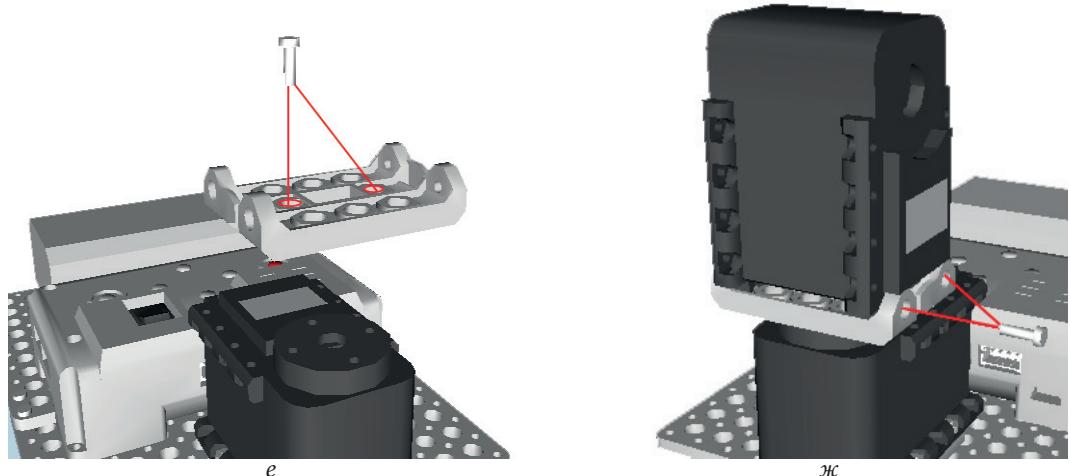


Рис. 8.2. Сборка следящей платформы. е - шаг 6, ж - шаг 7

Шаг 9. Соединяем проводами сервоприводы и контроллер, подключаем питание от аккумулятора. Платформа готова (рис. 8.2. (к)).

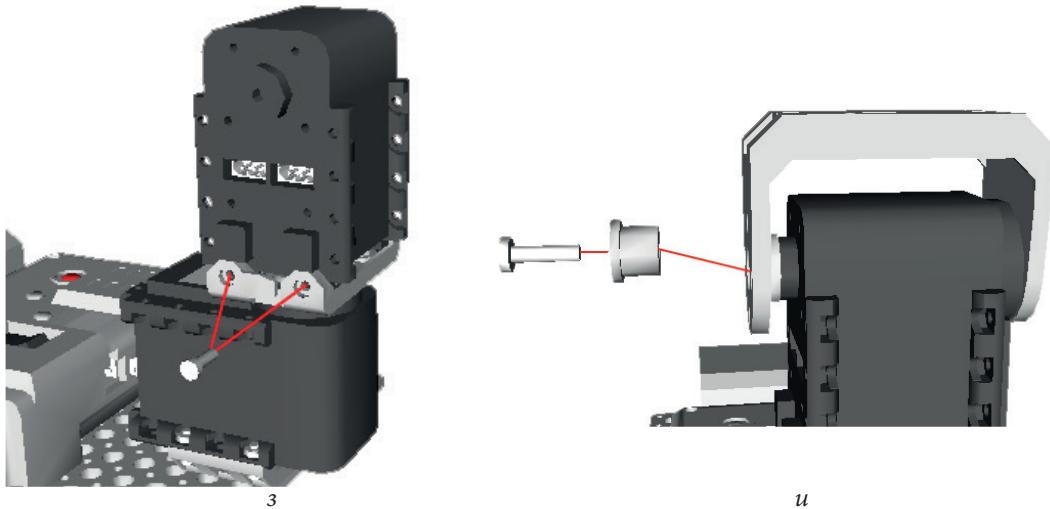


Рис. 8.2. Сборка следящей платформы. 3 - шаг 7, и - шаг 8.

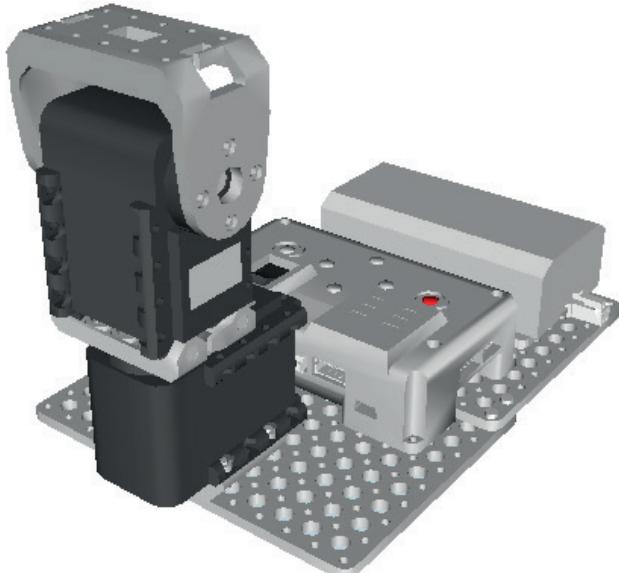


Рис. 8.2. (к). Сборка следящей платформы. Шаг 9.

На верхнюю часть платформы любым доступным способом крепится модуль TrackingCam любым удобным способом. Например, с помощью распечатанного на 3D принтере основания.

Основная задача pan and tilt платформы - обеспечить телекамере, в нашем случае модулю TrackingCam, возможность следить за определенным объектом, а именно всегда направлять объектив модуля точно на требуемый объект с помощью сервоприводов.

Чтобы система могла следить за объектом – оранжевым мячом – она должна обладать информацией о его текущем положении. В данном

случае, в качестве датчика положения объекта используется система технического зрения в виде модуля TrackingCam, который необходимо настроить на распознавание мяча. Для этого необходимо воспользоваться утилитой TrackingCam App, задать цветовые и морфологические признаки объекта, настроить параметры порта ввода-вывода информации и сохранить настройки как было разобрано ранее.

После настройки телекамеры приступим к настройке самой платформы. Поскольку каждый сервопривод DYNAMIXEL адресуется по уникальному в системе ID-номеру, зададим, например, нижнему сервоприводу ID равный 1, а верхнему ID равный 2. Сервоприводы нужно переключить в режим работы в качестве шарнира (Joint). Эти настройки выполняются в утилите RoboPlus Manager (рис. 8.3.), входящей в набор стандартных приложений RoboPlus.

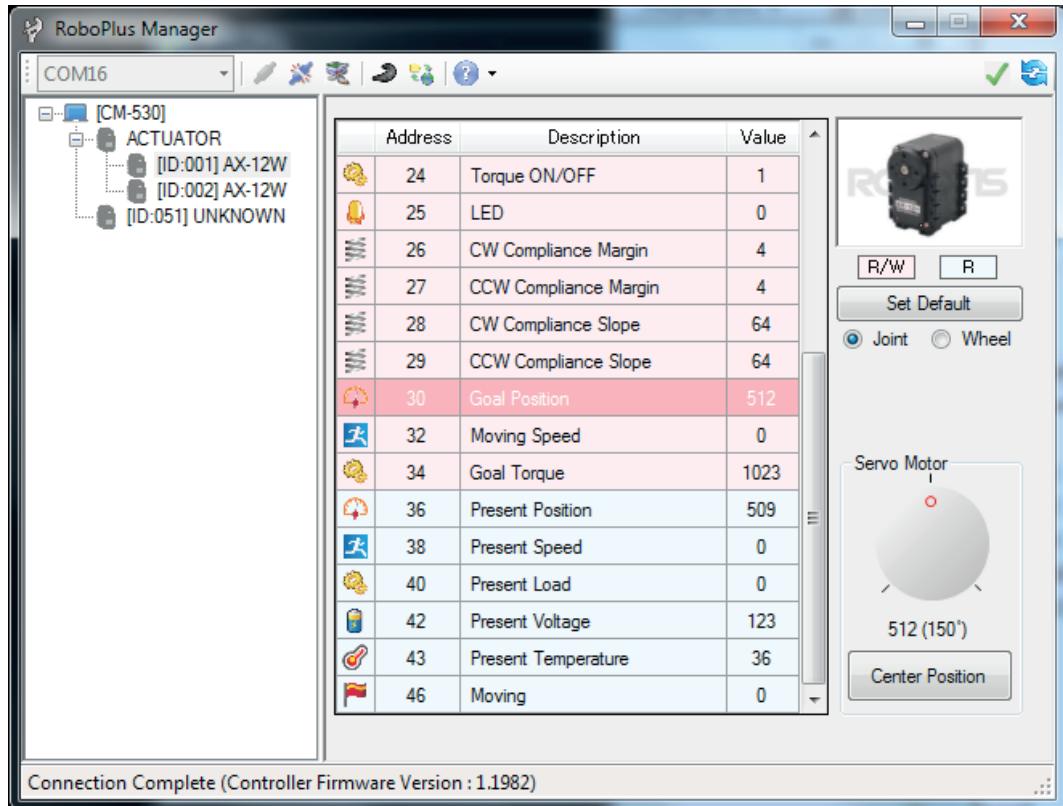


Рис. 8.3. Окно утилиты RoboPlus Manager

Запустим утилиту RoboPlus Manager. Подключим контроллер к компьютеру с помощью USB кабеля и включим его. Обратите внимание - для работы контроллера необходимо подать на него внешнее питание, либо от аккумулятора, либо от сети. В левом верхнем углу RoboPlus Manager необходимо выбрать СОМ-порт, под которым контроллер установился в системе и нажать на соседнюю кнопку - Connect.

Произойдет подключение утилиты к контроллеру, сопровождающееся звуковым сигналом. В случае, если подключение выполнилось успешно, содержимое окна RoboPlus Manager изменится и в левой части окна можно будет наблюдать подключенные к контроллеру DYNAMIXEL - совместимые устройства, а в центральной - параметры этих устройств. Для настройки сервоприводов необходимо выбрать нужный привод и указать ему в правой стороне окна режим работы - Joint.

Узнать настройки какого привода (верхнего или нижнего) редактируются в данный момент можно, изменив положение его вала. Для этого нужно выбрать параметр Goal Position и с помощью появившегося джойстика вращать привод. Затем нужно установить ему соответствующий параметр ID с помощью выпадающего списка в правом нижнем углу окна. После установки ID Нажать кнопку Apply. Для другого сервопривода действия следует повторить. После этого платформа готова к использованию.

Напишем управляющую программу. Для этого создадим новый проект в среде R+Task 2.0 под контроллер СМ- 530. Для опроса телекамеры добавим в программу функцию TrackingCamParseBlobs, подробно рассмотренную ранее. Эта функция является основной для работы с модулем технического зрения, и в дальнейшем все программы будут завязаны именно на ее использование. Для сокращения объема кода удалим из этой функции все строки, отвечающие за вывод данных на экран. В итоге внешний вид функции будет как на рисунке 8.4.

Реализуем алгоритм управления платформой. Слежение за объектом будет выполняться следующим образом. С модуля TrackingCam запрашиваются данные о распознанном объекте. Если он виден в центральной части изображения, то модуль наведен на объект и действий не требуется. Иначе происходит поворот модуля на фиксированный угол вокруг одной из осей в ту сторону, в которую мяч смешен относительно оптической оси телекамеры наибольшим образом.

Для реализации алгоритма управления добавим в тело главной функции START PROGRAMM программы бесконечный цикл ENDLESS LOOP. Внутри этого цикла мы и разместим ту часть кода, которая будет отвечать за проверку условий и управлять сервоприводами. Алгоритм, по которому будет выполняться слежение за объектом можно описать следующим образом: Происходит опрос данных о распознанной области с модуля TrackingCam. Если центр распознанной области находится в центральной зоне видимости модуля, то считается что модуль наведен на объект корректно и никаких действий не происходит. Если же

```

16 FUNCTION TrackingCamParseBlobs
17 {
18     TrackingCamID = 51
19     TrackingCamBlobStorageAddr = 16
20     TrackingCamEmptyBlobCode = 255
21     addr = TrackingCamBlobStorageAddr
22     LOOP FOR ( i = 0 ~ 15 )
23     {
24         BlobType = [ ID[TrackingCamID]: ADDR[addr(b)] ]
25         addr = addr + 2
26         IF ( BlobType == 255 || BlobType == -1 || BlobType == 248 )
27         {
28             BREAK LOOP
29         }
30         BlobCx = [ ID[TrackingCamID]: ADDR[addr(w)] ]
31         addr = addr + 2
32         BlobCy = [ ID[TrackingCamID]: ADDR[addr(w)] ]
33         addr = addr + 2
34         BlobArea = [ ID[TrackingCamID]: ADDR[addr(w)] ] * 4
35         addr = addr + 2
36         BlobLeft = [ ID[TrackingCamID]: ADDR[addr(w)] ]
37         addr = addr + 2
38         BlobRight = [ ID[TrackingCamID]: ADDR[addr(w)] ]
39         addr = addr + 2
40         BlobTop = [ ID[TrackingCamID]: ADDR[addr(w)] ]
41         addr = addr + 2
42         BlobBottom = [ ID[TrackingCamID]: ADDR[addr(w)] ]
43         addr = addr + 2
44     }
45 }

```

Рис. 8.4. Вид функции *TrackingCamParseBlobs*

центр области оказывается смещен относительно центральной части зоны видимости модуля, то происходит доворот модуля в соответствующую сторону. Поворот для компенсации смещения объекта вдоль оси X изображения будет осуществлять сервопривод с ID 1, а компенсации смещения вдоль оси Y сервопривод с ID 2. Эти команды будут выполняться следующим образом:

В первом случае осуществляется поворот сервопривода с ID 1 против часовой стрелки на 10 единиц (минимальных возможных углов поворота сервопривода), во втором случае этот же сервопривод поворачивает-

ся по часовой стрелке на 10 единиц. Аналогично и второй сервопривод в первом случае поворачивается вверх, во втором случае вниз.

Данные команды реализованы в среде R+Task с помощью инструкции COMPUTE.

<input type="checkbox"/> ID[1]: ⚡ Goal Position	=	<input type="checkbox"/> ID[1]: ⚡ Present Position	+ 10
<input type="checkbox"/> ID[1]: ⚡ Goal Position	=	<input type="checkbox"/> ID[1]: ⚡ Present Position	- 10
<input type="checkbox"/> ID[2]: ⚡ Goal Position	=	<input type="checkbox"/> ID[2]: ⚡ Present Position	+ 10
<input type="checkbox"/> ID[2]: ⚡ Goal Position	=	<input type="checkbox"/> ID[2]: ⚡ Present Position	- 10

Таким образом, при изменении положения распознанного объекта относительно оптической оси телекамеры будет осуществляться поворот каждого сервопривода в нужную сторону. Для этого создадим конструкцию из серии условных переходов с операторами IF - ELSE после вызова функции TrackingCamParseBlobs и для оператора в качестве условия укажем проверку положения BlobCx или BlobCy относительно границ центральной зоны видимости телекамеры (рис. 8.5.).

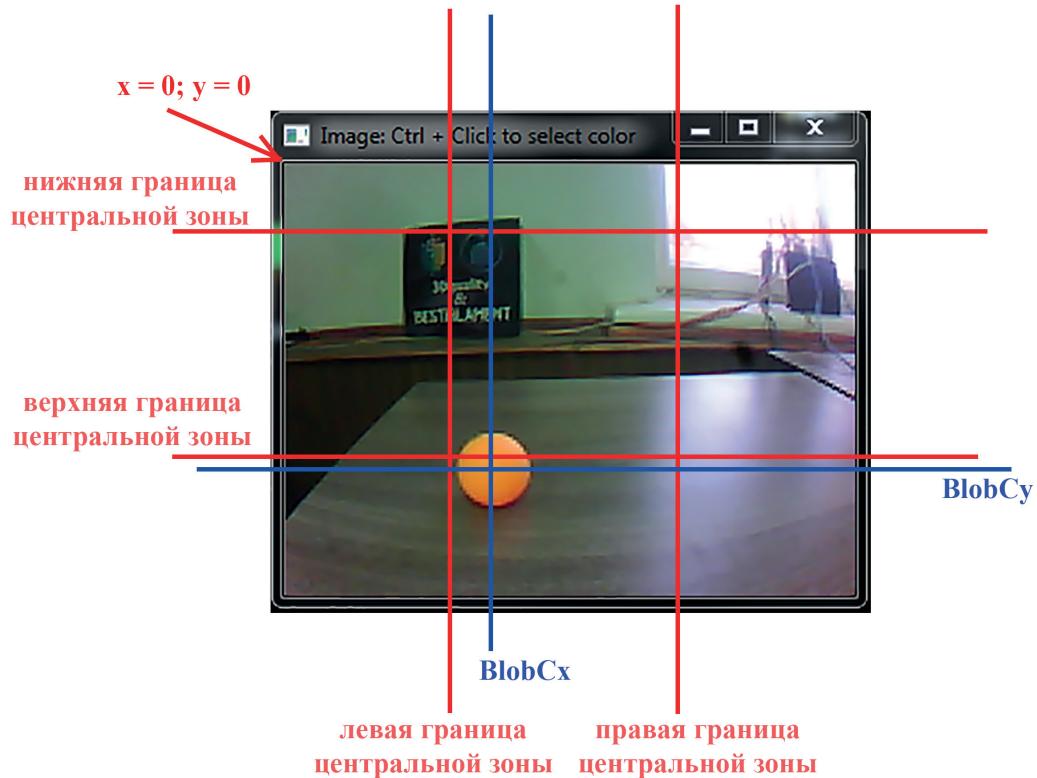


Рис. 8.5. Схема разметки области видимости TrackingCam

В данном примере в качестве условий было определено следующее - BlobCx сравнивается со значениями 106 и 212, а BlobCy - со значениями 80 и 160. В зависимости от условия в тело каждого условного перехода IF добавляем команду управления сервоприводами - если BlobCx меньше 106, то поворачиваем платформу против часовой стрел-

ки, если больше 212, то по часовой стрелке. Если BlobCx меньше 80 то поворачиваем платформу вверх, если больше 160, то поворачиваем платформу вниз. После последней инструкции ELSE добавим команды остановки сервоприводов. Это необходимо в том случае, если модуль потеряет объект из видимости. В результате получившийся код будет выглядеть следующим образом (рис. 8.6.):

```
10 IF ( BlobCx <= 106 )
11 {
12     ID[1]: ⚡ Goal Position = ID[1]: ⚡ Present Position + 10
13 }
14 ELSE IF ( BlobCx >= 212 )
15 {
16     ID[1]: ⚡ Goal Position = ID[1]: ⚡ Present Position - 10
17 }
18 IF ( BlobCy <= 80 )
19 {
20     ID[2]: ⚡ Goal Position = ID[2]: ⚡ Present Position - 10
21 }
22 ELSE IF ( BlobCy >= 160 )
23 {
24     ID[2]: ⚡ Goal Position = ID[2]: ⚡ Present Position + 10
25 }
26 ELSE
27 {
28     ID[1]: ⚡ Goal Velocity = 0
29     ID[2]: ⚡ Goal Velocity = 0
30 }
```

Рис. 8.6. Код управляющей программы, отвечающей за движение сервоприводов

Для удобства добавим перед бесконечным циклом ENDLESS LOOP две инструкции инициализации переменных LOAD с указанием установки каждого сервопривода в начальную позицию. Итоговый вид основной функции показан на рисунке 8.7.

Управляющая программа для следящей платформы готова. Если ее загрузить в контроллер платформы и запустить контроллер в автономном режиме, то модуль технического зрения TrackingCam будет всегда направлен на оранжевый мяч, как бы его не сместили относительно центра зоны видимости модуля.

```

1 START PROGRAM
2 {
3     ID[1]: ⚡ Goal Position = 512
4     ID[2]: ⚡ Goal Position = 512
5 ENDLESS LOOP
6 {
7     CALL TrackingCamParseBlobs
8     ⏳ High-resolution Timer = 0.033sec
9     WAIT WHILE ( ⏳ High-resolution Timer > 0.000sec )
10    IF ( BlobCx <= 106 )
11    {
12        ID[1]: ⚡ Goal Position = ID[1]: ⚡ Present Position + 10
13    }
14    ELSE IF ( BlobCx >= 212 )
15    {
16        ID[1]: ⚡ Goal Position = ID[1]: ⚡ Present Position - 10
17    }
18    IF ( BlobCy <= 80 )
19    {
20        ID[2]: ⚡ Goal Position = ID[2]: ⚡ Present Position - 10
21    }
22    ELSE IF ( BlobCy >= 160 )
23    {
24        ID[2]: ⚡ Goal Position = ID[2]: ⚡ Present Position + 10
25    }
26    ELSE
27    {
28        ID[1]: ⚾ Goal Velocity = 0
29        ID[2]: ⚾ Goal Velocity = 0
30    }
31 }
32 }

```

Рис. 8.7. Основная функция управляющей программы

8.2 Следование вдоль сложной линии

Одной из типовых соревновательных задач, связанных со следованием вдоль линии является задача следования вдоль линии, состоящей из отдельных объектов разной формы, которые могут быть так же и разного цвета. Такая задача до недавнего времени являлась задачей для самых старших участников, так как ее решение требовало достаточно глубоких знаний в области работы с одноплатными компьютерами, операционной системой Linux и специализированными библиотеками для обработки видео сигнала, такими как OpenCV. Такой высокий порог входления в эти соревнования отсеивал большой процент участников, желающих принять в нем участие, в первую очередь школьников, не имеющих должной базовой подготовки. Использование готового модуля TrackingCam в настоящее время сильно понижает порог входления в данные соревнования, и позволяет принимать участие всем желающим.

В данном примере мы рассмотрим процесс подготовки платформы и разработки управляющей программы для автоматического следования платформы вдоль сложной линии, ориентируясь на ней с помощью TrackingCam. В данном примере мы рассмотрим следующие вопросы:

1. Сборка и подготовка типовой платформы
2. Настройка модуля технического зрения TrackingCam
3. Разработка управляющей программы для движения вдоль сложной линии
4. Проверка системы на практике

Сборка и подготовка типовой платформы

Для решения задачи следования по сложной линии достаточно использовать какую-либо типовую платформу, для удобства – дифференциального типа. Выбор такой платформы облегчит задачу разработки управляющей программы, т.к. в случае двухколесной дифференциальной платформы описание кинематики будет довольно простым. Таким образом соберем платформу, подобную показанной на рисунке 8.8.

В качестве управляющего контроллера воспользуемся arduino-подобной аппаратной платформой OpenCM, установленной на плате расширения Expansion Board 485, либо же на Stem Board. Питание обеспечим с помощью стандартного LiPo аккумулятора, выдающегоnomинальное напряжение в 11.1В.

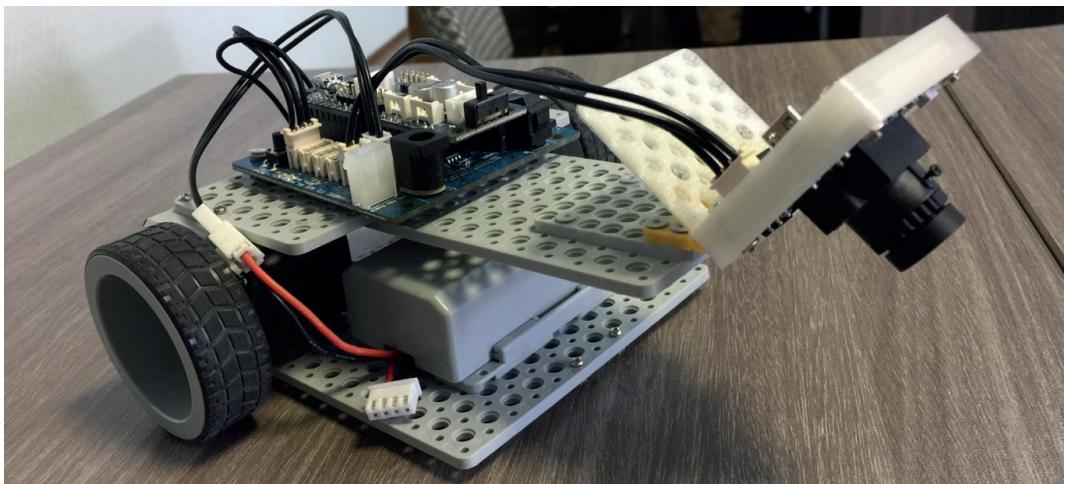


Рис. 8.8. Пример внешнего вида платформы для следования вдоль сложной линии

Соединение сервоприводов рекомендуется выполнить последовательно и подключить получившуюся цепь к плате расширения Expansion Board.

Подключение модуля технического зрения TrackingCam выполняется с помощью стандартного 3х пинового Dynamixel кабеля к любому из свободных портов платы расширения.

Перед тем как приступить к разработке управляющей программы необходимо уточнить ID сервоприводов, установленных на роботе. Для удобства условимся, что левый сервопривод имеет ID равным 1, а правый сервопривод имеет ID равный 2. Для того, чтобы уточнить и изменить сервопривод можно воспользоваться способом, рассмотренным в примере разработки следящей платформы. В отличии от следящей платформы, где сервоприводы использовались в режиме шарнира (сочленения) в данном примере сервоприводы должны использоваться в режиме колеса. Однако, поскольку в отличии от предыдущего примера здесь управляющим контроллером является OpenCM, являющийся более продвинутым по сравнению с CM530, то данный режим работы будет устанавливаться напрямую в коде управляющей программы.

Отличием от предыдущего примера, в котором модуль технического зрения отслеживал только одну область, и, соответственно, выдавал информацию только об одной области, в данном примере модуль должен отслеживать сразу несколько областей, соответствующих одному шаблону – в нашем случае одному цвету. Линия, вдоль которой необходимо двигаться роботу в данном случае представлена разрозненными черными фигурами разной формой – прямоугольниками со скругленными краями и кругами. Для отслеживание этих фигур на-

строим модуль по аналогии со случаем отслеживание одной однотонной области. Поскольку линия представлена хоть и разными фигурами, но имеющими один цвет, то мы не будем задаваться вопросом их формы – в данном случае это для нас не важно. Изображения с камеры модуля и результат обработки отдельных элементов траектории представлены на рисунках 8.9 и 8.10:

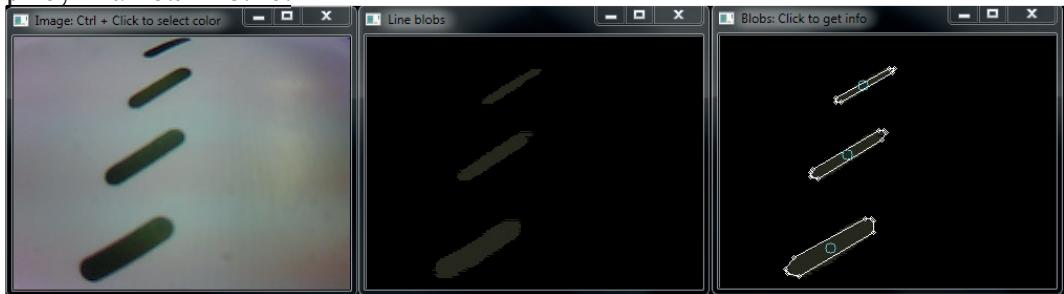


Рис. 8.9. Часть траектории из прямоугольников

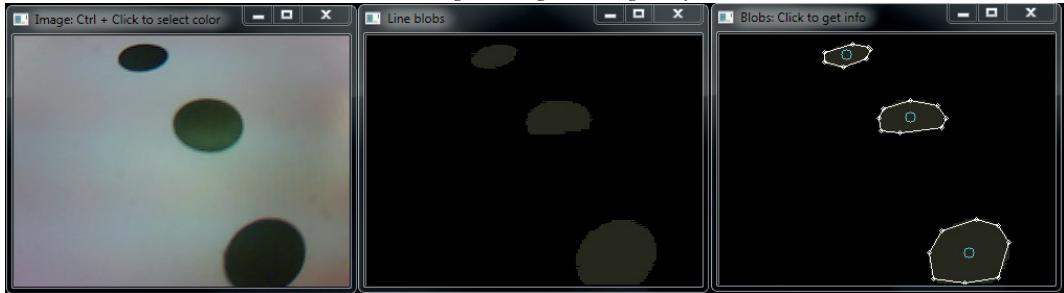


Рис. 8.10. Часть траектории из кругов

Как видно на представленных изображениях в кадр попадает сразу несколько распознаваемых областей, и, соответственно необходимо в дальнейшем для лучшего движения вдоль такой линии использовать информацию о всех областях, распознанных на одном кадре.

Сохраним настройки модуля в его память и перейдем к разработке управляющей программы.

В модели данного робота в качестве управляющего контроллера используется Arduino-подобный контроллер OpenCM. Для работы с ним используется среда разработки Robotis IDE. Разработку управляющей программы разобьем на 2 этапа:

на первом этапе напишем необходимые функции для движения робота вперед и поворотов, на втором добавим в программу работу с модулем технического зрения.

Для инициализации соединения для обмена данными по протоколу Dynamixel в коде программы необходимо указать шину – другими словами порты, на плате, к которым будет выполнено подключение сервоприводов и модуля технического зрения. Далее необходимо указать

ID сервоприводов, которыми необходимо управлять. Все это делается в коде программы до функции `setup()` следующими командами:

```
#define DXL_BUS_SERIAL3 3  
#define ID_NUM_1 1  
#define ID_NUM_2 2  
Dynamixel Dxl(DXL_BUS_SERIAL3);
```

в функции `setup()` производится непосредственно инициализация соединения командой

```
Dxl.begin(3);
```

и задание режима работы сервоприводов в качестве колес командами

```
Dxl.wheelMode(ID_NUM_1);  
Dxl.wheelMode(ID_NUM_2);
```

Затем создадим основные функции, отвечающие за движение робота – `forward`, `left`, `right`, `stop`. Разместить их необходимо после функции `loop`. Функция движения вперед будет выглядеть следующим образом:

```
void forward (){  
Dxl.goalSpeed(ID_NUM_1, 400);  
Dxl.goalSpeed(ID_NUM_2, 400 | 0x400);  
}
```

здесь команда `goalSpeed` означает установку определенной скорости и она имеет 2 аргумента – ID сервопривода и его скорость. В случае с сервоприводом с ID `ID_NUM_2` мы учитываем зеркальную относительно сервоприводу с ID 1 установку сервопривода, в результате чего задаем ему вращение в обратную сторону.

Аналогичным образом напишем функции для поворотов налево, направо и остановки:

налево:

```
void left (){  
Dxl.goalSpeed(ID_NUM_1, 400 | 0x400);  
Dxl.goalSpeed(ID_NUM_2, 400 | 0x400);  
}
```

направо:

```
void right (){  
Dxl.goalSpeed(ID_NUM_1, 400 );  
Dxl.goalSpeed(ID_NUM_2, 400 );  
}
```

СТОП:

```
void stop() {  
    Dxl.goalSpeed(ID_NUM_1, 0);  
    Dxl.goalSpeed(ID_NUM_2, 0);  
}
```

Для проверки правильности движения робота в функцию loop добавим вызов полученных функций в определенном порядке:

```
void loop() {  
    forward();  
    delay(2000);  
    right();  
    delay(2000);  
    left();  
    delay(2000);  
    stop();  
    delay(5000);  
}
```

после загрузки кода в контроллер робот должен начать движение вперед и двигаться в течении 2х секунд, затем вращаться по часовой стрелке в течении 2х секунд и против часовой стрелки так же в течении двух секунд. После этого робот должен остановится на 5 секунд. Если робот повел себя именно так, значит платформа собрана верно и команды управления движением указаны так же верно. На текущем этапе код управляющей программы должен выглядеть следующим образом:

```
#define DXL_BUS_SERIAL3 3  
#define ID_NUM_1 1  
#define ID_NUM_2 2  
Dynamixel Dxl(DXL_BUS_SERIAL3);  
  
void setup() {  
    Dxl.begin(3);  
    Dxl.wheelMode(ID_NUM_1);  
    Dxl.wheelMode(ID_NUM_2);  
}  
  
void loop() {  
    forward();  
    delay(2000);  
    right();  
    delay(2000);  
    left();  
    delay(2000);  
    stop();  
    delay(5000);
```

```

}

void forward (){
Dxl.goalSpeed(ID_NUM_1, 400);
Dxl.goalSpeed(ID_NUM_2, 400 | 0x400);
}

void left (){
Dxl.goalSpeed(ID_NUM_1, 400 | 0x400);
Dxl.goalSpeed(ID_NUM_2, 400 | 0x400);
}

void right (){
Dxl.goalSpeed(ID_NUM_1, 400 );
Dxl.goalSpeed(ID_NUM_2, 400 );
}

void stop() {
Dxl.goalSpeed(ID_NUM_1, 0);
Dxl.goalSpeed(ID_NUM_2, 0);
}

```

На этом разработка первой части управляющей программы закончена. Приступим к написанию кода для работы с модулем технического зрения.

Добавим в самое начало программы код, отвечающий за инициализацию структуры, необходимой нам в дальнейшем:

```

struct TrackingCamBlobInfo_t
{
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint32 area;
    uint16 left;
    uint16 right;
    uint16 top;
    uint16 bottom;
};

```

после нее проинициализируем массив, в котором будут хранится данных о всех распознанных областях:
`TrackingCamBlobInfo_t blob[15];`

Так же укажем ID камеры и проинициализируем переменную для хранения адреса считывания данных:

```

#define ID_NUM 51
uint16 addr = 16;

```

Для получения данных с модуля технического зрения о всех распознанных областях поместим в самый конец программы функцию работы с камерой, подробно разобранную выше в соответствующем примере:

```
void TrackingCamGetBlobs (){
    uint16 curAddr = addr;
    for (uint8 i = 0; i < 16; i++){
        SerialUSB.print(i);
        SerialUSB.print(" ");
        blob[i].type = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].type);
        SerialUSB.print(" ");
        if (blob[i].type == 255 || blob[i].type == -1 || blob[i].type == 248){
            SerialUSB.println(" ");
            break;
        }
        blob[i].cx = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].cx);
        SerialUSB.print(" ");
        blob[i].cy = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].cy);
        SerialUSB.print(" ");
        blob[i].area = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].area);
        SerialUSB.print(" ");
        blob[i].left = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].left);
        SerialUSB.print(" ");
        blob[i].right = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].right);
        SerialUSB.print(" ");
        blob[i].top = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.print(blob[i].top);
        SerialUSB.print(" ");
        blob[i].bottom = Dxl.readByte(ID_NUM, curAddr);
        curAddr = curAddr + 2;
        SerialUSB.println(blob[i].bottom);
    }
}
```

Теперь перейдем к самому алгоритму движения. Поскольку в область видимости камеры попадает сразу несколько объектов, которые

можно использовать для определения характера (изгиба) траектории, то можно придумать некий хитрый алгоритм, который будет это учитывать. В нашем же примере для навигации мы будем использовать только один (ближайший) объект, на который и будет ехать робот. Поскольку мы всегда может получить его центр (с помощью переменной blob[i].cx, то при движении будем проверять нахождение центра этого объекта в определенных рамках. Для удобства разобьем область видимости камеры по оси x на 3 равные части и получим границы нашей области – 106 и 213 пикселей.

Внутри функции loop() заменим все ранее написанное на следующий код:

```
TrackingCamGetBlobs();
delay(30);
if (blob[0].cx < 106)
left();
else if (blob[0].cx > 213)
right();
else
forward();
```

здесь происходит вызов функции получения данных с модуля технического зрения, выполняется небольшая задержка, после чего выполняется анализ – если центр первой области находится левее центральной области сцены, то происходит поворот налево, если правее – направо. Если центр области находится пределах центральной зоны, то робот движется прямо.

На этом разработка программы для движения робота вдоль линии окончена. Полный код программы представлен ниже:

```
struct TrackingCamBlobInfo_t
{
    uint8 type;
    uint16 cx;
    uint16 cy;
    uint32 area;
    uint16 left;
    uint16 right;
    uint16 top;
    uint16 bottom;
};

TrackingCamBlobInfo_t blob[15];

#define DXL_BUS_SERIAL3 3
```

```
#define ID_NUM_1 1
#define ID_NUM_2 2
#define ID_NUM 51

uint16 addr = 16;

Dynamixel Dxl(DXL_BUS_SERIAL3);

void setup() {
  Dxl.begin(3);
  Dxl.wheelMode(ID_NUM_1);
  Dxl.wheelMode(ID_NUM_2);

}

void loop() {
  TrackingCamGetBlobs();
  delay(30);
  if (blob[0].cx < 106)
    left();
  else if (blob[0].cx > 213)
    right();
  else
    forward();
}
void forward (){
  Dxl.goalSpeed(ID_NUM_1, 100);
  Dxl.goalSpeed(ID_NUM_2, 100 | 0x400);
}

void left (){
  Dxl.goalSpeed(ID_NUM_1, 100 | 0x400);
  Dxl.goalSpeed(ID_NUM_2, 100 | 0x400);
}

void right (){
  Dxl.goalSpeed(ID_NUM_1, 100 );
  Dxl.goalSpeed(ID_NUM_2, 100 );
}

void stop() {
  Dxl.goalSpeed(ID_NUM_1, 0);
  Dxl.goalSpeed(ID_NUM_2, 0);
}

void TrackingCamGetBlobs (){
  uint16 curAddr = addr;
  for (uint8 i = 0; i < 16; i++){
    SerialUSB.print(i);
    SerialUSB.print(" ");
  }
}
```

```
blob[i].type = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].type);
SerialUSB.print(<< <<);
if (blob[i].type == 255 || blob[i].type == -1 || blob[i].type == 248){
SerialUSB.println(<< <<);
break;
}
blob[i].cx = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].cx);
SerialUSB.print(<< <<);
blob[i].cy = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].cy);
SerialUSB.print(<< <<);
blob[i].area = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].area);
SerialUSB.print(<< <<);
blob[i].left = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].left);
SerialUSB.print(<< <<);
blob[i].right = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].right);
SerialUSB.print(<< <<);
blob[i].top = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.print(blob[i].top);
SerialUSB.print(<< <<);
blob[i].bottom = Dxl.readByte(ID_NUM, curAddr);
curAddr = curAddr + 2;
SerialUSB.println(blob[i].bottom);
}
}
```

Проверка системы на практике.

После загрузки программы в робота и установки его на поле робот будет двигаться вдоль траектории, отмеченной черными метками. Но при тестировании на реальном поле сразу возникают моменты, на которые обязательно необходимо обращать внимание.

Во-первых это расположение и сила внешнего источника освещения. Для корректного распознавания всех меток на всем поле необходимо чтобы поле находилось в области равномерного освещения. Если в одной области поля будет засветы или, наоборот, тени, то модуль технического зрения может из-за этого некорректно распознавать мет-

ки.

Во-вторых, это опять же связано с освещением, если на поле будут светить яркие точечные источники освещения, то на поле будет падать тень от самого робота, и в некоторых ситуациях, таких как при нахождении робота спиной к источнику освещения, модуль может некорректно распознавать метки из-за падающей на белое поле тени.

В-третьих – необходимо понимать, что процесс получения данных от модуля не мгновенный и требует некоторого времени. Из за этого при определенной скорости движения может возникнуть ситуация, при которой основной контроллер робота не будет успевать получать актуальные данные от модуля технического зрения и в результате робот будет перемещаться некорректно.

Таким образом подобное решение можно использовать при реализации движения вдоль траектории обозначенных разными объектами и даже объектами разных цветов. Такие задачи часто встречаются в различных робототехнических соревнованиях, ориентированных на продвинутых участников и использование серьезного аппаратного и программного обеспечения.

Заключение

В данном учебном пособии были рассмотрены устройство и функционал модуля технического зрения TrackingCam, а так же показана его применимость для решения задач соревновательного и научно-исследовательского направления.

Используя показанный функционал и возможности модуля TrackinCam, предложенные для разбора задачи можно усложнять путем повышения уровня проработки вопроса, либо же использования вычислительных компонентов и сред разработки сторонних производителей, что дает возможность принять данных модуль при решении задач не привязываясь к определенному набору электронных компонентов.

Подобный подход позволит применять данных модуль для участия в соревновательной деятельности различного направления от школьного до старшего возраста.

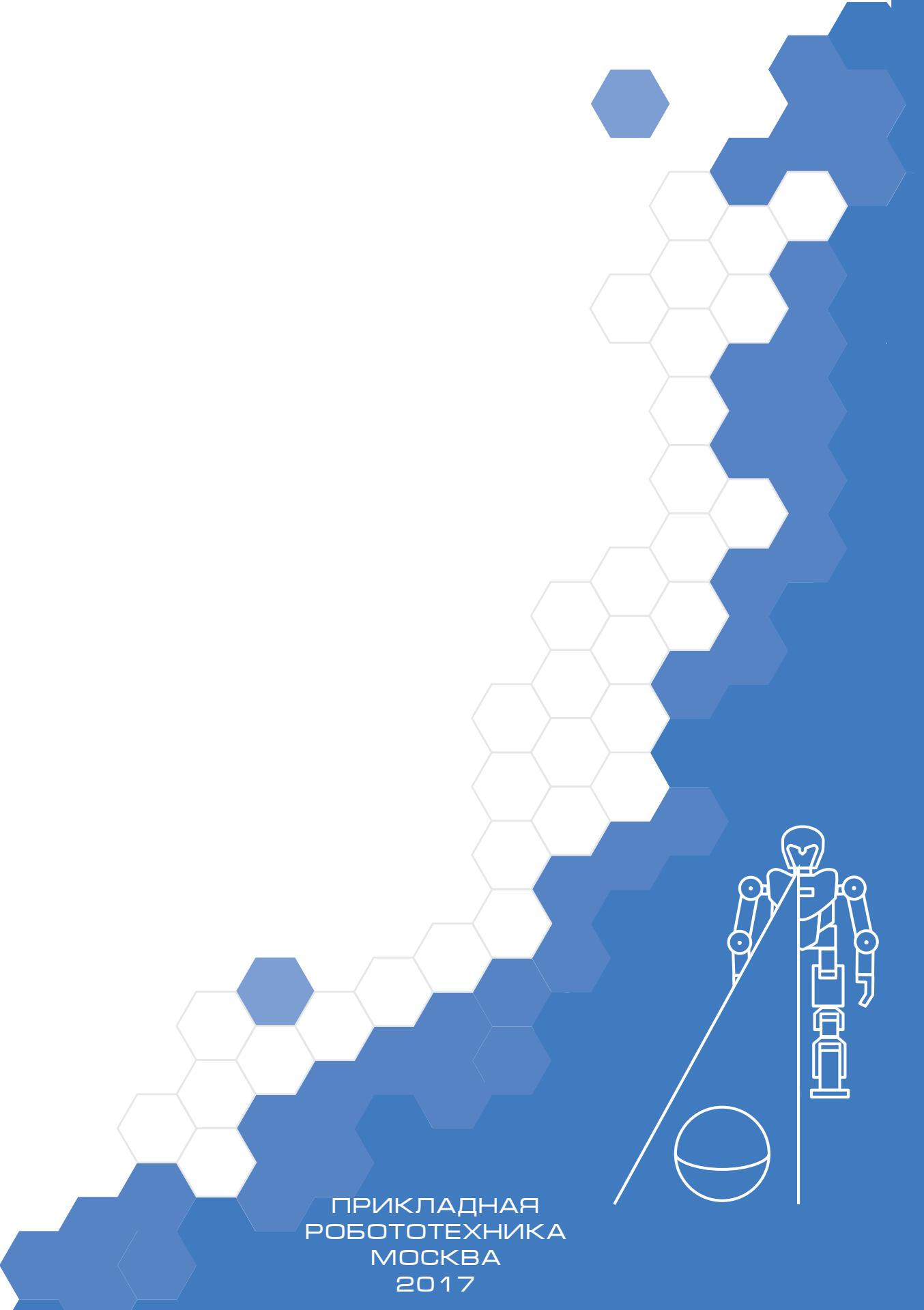


74





75



ПРИКЛАДНАЯ
РОБОТОТЕХНИКА
МОСКВА
2017

