Отзыв

на бакалаврскую работу студента 4 курса направления подготовки «Электроника и наноэлектроника» Кирсанова Александра Евгеньевича на тему «Разработка программного обеспечения роботизированной платформы предназначенной для обследования помещений»

Современные технические комплексы различного рода всё чаще дополняются элементами робототехнических систем. Одним из основных элементов является техническое зрение. На основе технического зрения роботизированные комплексы приобретают способность различать объекты реального мира и ориентироваться в пространстве. Задачи технического зрения являются сложными и многогранными, и поэтому разработка новых подходов к обработке визуальной информации всегда является актуальной задачей. Решением такого рода задачи и посвящена бакалаврская работа А. Е. Кирсанова.

В данной работе основной задачей является разработка программного обеспечения для автономной робототехнической платформы, предназначенного для управления платформой на основе визуальной информации, поступающей от видеокамеры робота.

С поставленной задачей Кирсанов А. Е. успешно справился. В результате был разработан программный код в среде LabVIEW для персонального компьютера управляющий автономным роботом Cymeroid. Суть управления заключается в том, что робот, ориентируясь по изображению окружающего пространства, должен проехать заранее заданный путь. Работоспособность кода была подтверждена экспериментом.

Проведенная работа свидетельствует о хороших знаниях дипломника основ программирования и электронной техники.

В целом, считаю, что проект Кирсанов Александра Евгеньевича заслуживает оценки «отлично», а выпускник Кирсанов А. Е. – присвоения квалификации бакалавра по направлению «Электроника и наноэлектроника».

Руководитель, доцент
кафедры электроники и наноэлектроники  М. В. Ильин

Заявление о самостоятельном характере выполнения

выпускной квалификационной работы

Я, Кирсанов Александр Евгеньевич, студент 4 курса, направления подготовки 11.03.04 «Электроника и наноэлектроника» заявляю, что в моей выпускной квалификационной работе на тему «Разработка программного обеспечения роботизированной платформы, предназначенной для обследования помещений», представленной в Государственную экзаменационную комиссию для публичной защиты, не содержится элементов неправомерных заимствований.

Все прямые заимствования из печатных и электронных источников, а также ранее защищённых письменных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим в Университете Положением о проверке выпускных квалификационных работ студентов ФГБОУ ВПО «МГУ им. Н. П. Огарёва» на наличие заимствований, в соответствии с которым обнаружение неправомерных заимствований является основанием для неудовлетворительной оценки выпускной квалификационной работы.

Подпись студента  Дата 11.06.2018

Работа представлена для проверки в Системе

Дата преставления ВКР *подпись руководителя ВКР*

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Н.П. ОГАРЁВА»
(ФГБОУ ВО «МГУ им. Н.П. Огарёва»)

ОТЧЕТ

о результатах проверки бакалаврской работы обучающегося на наличие заимствований

Автор работы: Кирсанов Александр Евгеньевич
Тема работы: «Разработка программного обеспечения роботизированной платформы, предназначенной для обследования помещений»
Руководитель: Ильин Михаил Владимирович

Представленная работа прошла проверку на наличие заимствований в системе «Антиплагиат. ВУЗ»

Результаты автоматической проверки:	оригинальность	93,65%
	цитирования	0%
	заимствования	6,35%

Результаты анализа полного отчета на наличие заимствований:

правомерные заимствование: есть
корректные цитирования: нет
неправомерные заимствования: нет
признаки обхода системы: нет

Общее заключение об итоговой оригинальности работы и возможности ее допуска к защите: высокая степень оригинальности текста, работа допущена к защите.

Руководитель
доцент кафедры
электроники и наноэлектроники М. В. Ильин

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н.П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра электроники и наноэлектроники

УТВЕРЖДАЮ

Заведующий кафедрой
к.т.н.

Н. Н. Беспалов

«22» 06 2018 г.

БАКАЛАВРСКАЯ РАБОТА

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РОБОТИЗИРОВАННОЙ ПЛАТФОРМЫ ПРЕДНАЗНАЧЕННОЙ ДЛЯ ОБСЛЕДОВАНИЯ ПОМЕЩЕНИЙ

Автор бакалаврской работы 15.06.2018 А. Е. Кирсанов

Обозначение бакалаврской работы БР – 02069964 – 11.03.04 – 08 – 18

Направление подготовки 11.03.04 электроника и наноэлектроника

Руководитель работы
доцент 16.06.18 М. В. Ильин

Нормоконтролер 20.06.2018 А. А. Шестёркина

Саранск
2018

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИСЛЕДОВАТЕЛЬСКИЙ МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н. П. ОГАРЁВА»

ИНСТИТУТ ЭЛЕКТРОНИКИ И СВЕТОТЕХНИКИ

КАФЕДРА ЭЛЕКТРОНИКИ И НАНОЭЛЕКТРОНИКИ

УТВЕРЖДАЮ

Заведующий кафедрой, к.т.н.

 Н. Н. Беспалов

«12» 10 2017 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Кирсанов Александр Евгеньевич

1 Тема: Разработка программного обеспечения робототизированной платформы предназначенной для обследования помещений.

Утверждена приказом по МордГУ № 8241-с от 12.10.2017 г.

2 Срок представления работы к защите 15 июня 2018 г.

3 Исходные данные для выпускной квалификационной работы:

– программное обеспечение предназначено для управления робототизированной автономной платформой Cymeroid;

– входная информация для программного обеспечения: траектория движения платформы; видеопоток, полученный с видеокамеры автономной платформы;

– программный код, обрабатывающий видеопоток должен выполняться на удалённом компьютере;

– программное обеспечение на основе известного пути следования платформы и её текущих координат должно вырабатывать управляющие воздействие для автономной платформы;

– текущие координаты должны определяться на основе видеоданных;

– среда разработки программного обеспечения LabVIEW 2012;

– модуль для обработки видео потока NI Vision.

4 Содержание выпускной квалификационной работы.

4.1 Введение.

4.2 Анализ методов определения координат и направления движения объектов на основе видеоданных.

4.3 Разработка алгоритма и программного кода определения координат автономной платформы.

4.4 Разработка алгоритма и программного кода определения направления движения автономной платформы.

4.5 Экспериментальное подтверждение работоспособности алгоритмов.

4.6 Заключение.

4.7 Список использованных источников.

5 Приложения

5.1 Программный код.

Руководитель работы М. В. Ильин

Задание принял к исполнению А. Е. Кирсанов

РЕФЕРАТ

Пояснительная записка содержит 52 листа, 17 рисунков, 2 таблицы, 5 источников.

CYMEROID, LABVIEW, РОБОТИЗИРОВАННАЯ ПЛАТФОРМА, WI-FI, АЛГОРИТМ ЛУКАСА И КАНАДЕ, АЛГОРИТМ ХОРНА И ШУНКА, TCP/IP.

Объектом разработки является программное обеспечение.

Цель работы – разработка программного обеспечения роботизированной платформы, предназначенной для обследования помещений.

Степень внедрения – частичная, в производственных помещениях и пищевых складов.

Область применения – данный программный комплекс позволяет сканирование помещений, в частности пола и нижних частей стен.

Значимость работы – разработанное программное обеспечение может использоваться при обследовании жилых и производственных помещений специальной роботизированной платформой.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящее время характеризуется стремительным появлением и развитием новых информационных технологий. Одной из таких новых и революционных технологий является технология виртуальных приборов, позволяющая создавать системы измерения, управления и диагностики различного назначения практически любой производительности и сложности. Суть этой технологии состоит в том, что измерительная и управляющая часть приборов и систем реализуется на аппаратной основе (устройств ввода-вывода аналоговых и цифровых сигналов), а их функциональная часть и пользовательский интерфейс – программными способами. Преимущество и эффективность виртуальных измерительных технологий состоит в возможности программным путем, опираясь на мощь современной компьютерной техники, создавать разнообразные приборы, измерительные системы и программно-аппаратные комплексы, легко перестраивать их к изменяющимся требованиям, уменьшить материальные затраты и время на разработку. При этом создаваемая измерительная система может быть оптимальным образом адаптирована для решения поставленных задач с учетом их особенностей.

Большие возможности по созданию виртуальных приборов предоставляет сегодня среда программирования LabVIEW (Laboratory Virtual Instrument Engineering Workbench) фирмы National Instruments, имеющая обширный инструментарий для создания программного обеспечения автоматизированных систем измерения и управления, решения задач технического зрения, создания распределенных приложений, Интернет-программирования и много другого.

1 Функциональная схема прототипа

В соответствии с техническим заданием, для управления роботизированной платформой необходимо установиться связь между PC-совместимым компьютером и роботизированной платформой. Связь осуществляется по беспроводной сети с помощью Wi-Fi модулей. Функциональная схема изображена на рисунке 1.

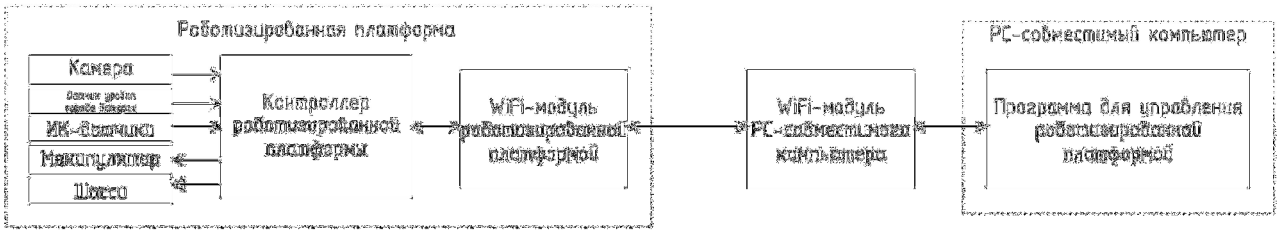


Рисунок 1 — Функциональная схема прототипа

2 Описание установки

Учебно-исследовательская платформа (УИП) «Cymeroid» предлагается в качестве универсальной технической и учебно-методической базы для практических занятий при обучении студентов основам робототехники, а также для разработки новых приложений по робототехнике. Главным устройством УИП является роботизированная платформа «Саймероид». Система разработана на аппаратной платформе NI sbRIO компании National Instruments, с использованием программным обеспечением, разработанным с использованием технологии виртуальных приборов в графической среде программирования NI LabVIEW.

Предлагаемая система включает в свой состав платформу для передвижения по поверхности, манипулятор с четырьмя степенями свободы для позиционирования исполнительного органа в трёхмерном пространстве, механический захват, монохромную видеокамеру, расположенную на манипуляторе, систему датчиков (включая дальномеры), Wi-Fi модуль и ряд вспомогательных принадлежностей, а также гибкое и наращиваемое программное обеспечение.

Роботизированная платформа представлена на рисунке 2.

Рисунок 2 — Роботизированная платформа Cymeroid

3 Описание проекта

Скомпилированный проект представлен на рисунке 3. В состав проекта входят несколько подпрограмм, файлы настроек, а также исходный код для прошивки роботизированной платформы.

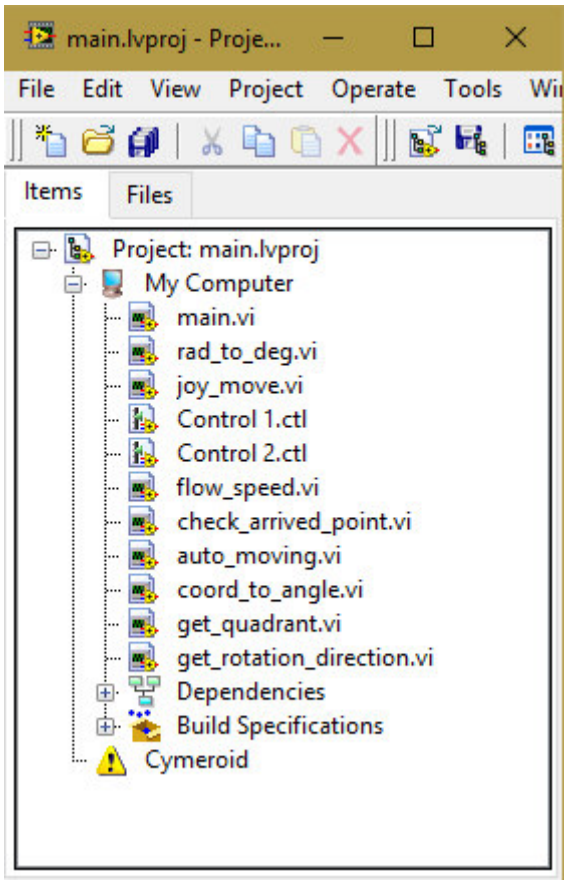


Рисунок 3 — Окно проекта

4 Описание системы управления

Разработка программного обеспечения ведётся в программном комплексе «LabView». Программный код состоит из двух параллельных циклов. В первом цикле реализована обработка изображения и логика управления роботизированной платформой. Во втором цикле реализована работа с передачей данных между PC-совместимым компьютером и роботизированной платформой. Блок-схема расположена на рисунке 4.

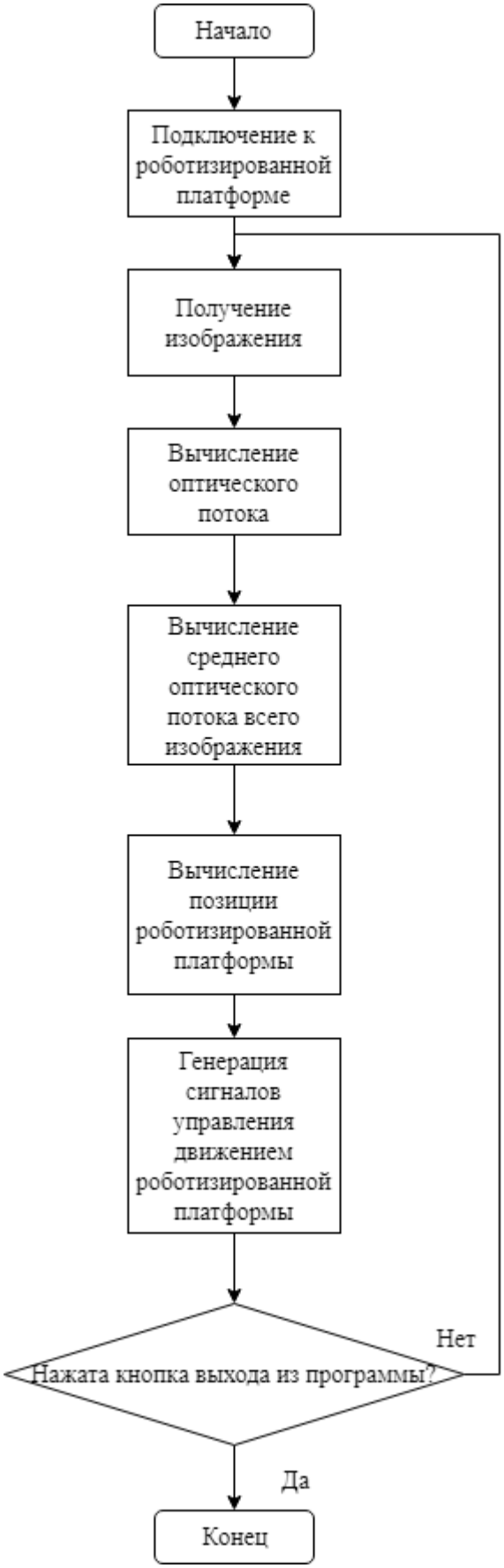


Рисунок 4 — Блок-схема программы

Общий вид программы представлен на рисунке 5.

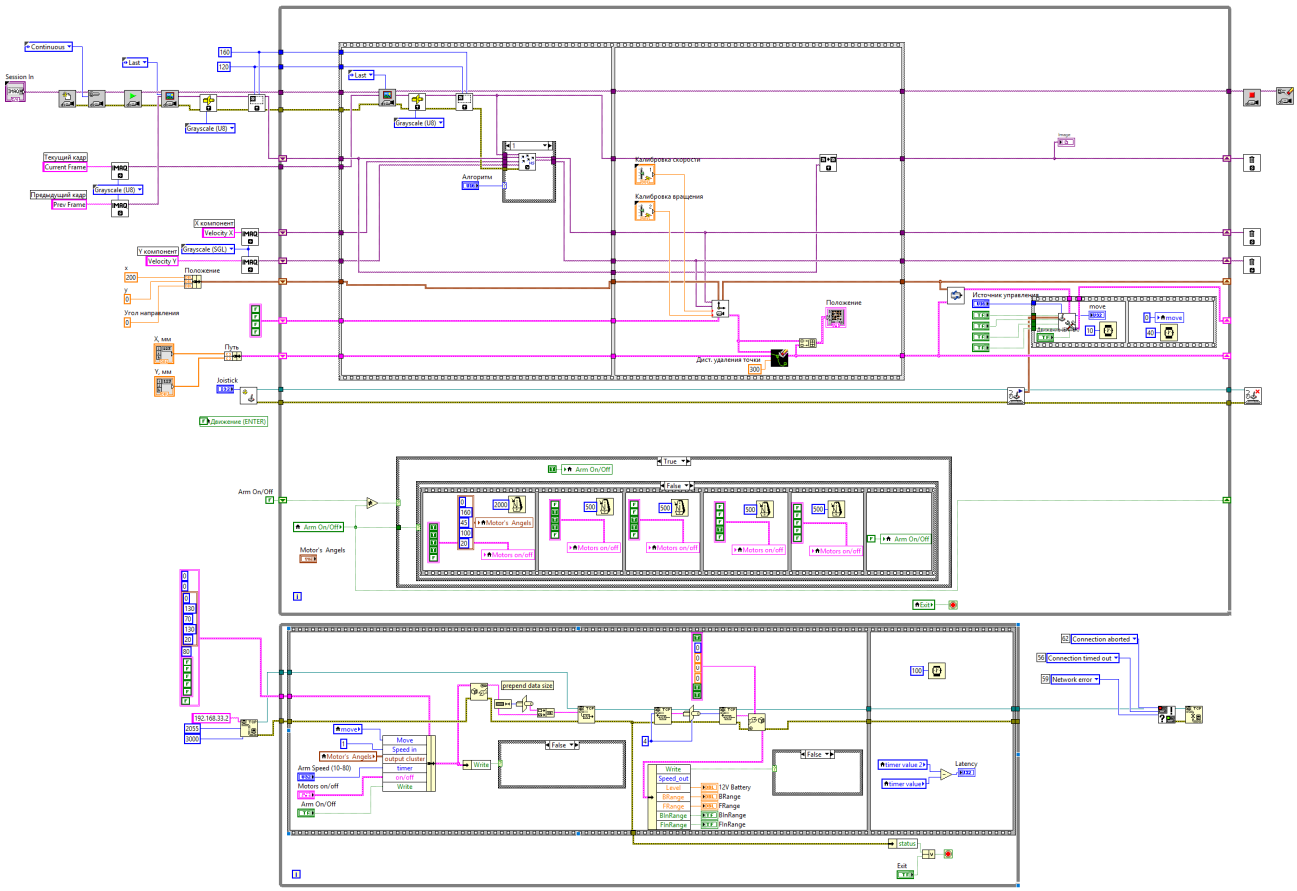


Рисунок 5 — Общий вид программы

Во время запуска программы происходит инициализация. С помощью подпрограмм IMAQdx Open Camera, IMAQdx Configure Acquisition, IMAQdx Start Acquisition реализована настройка канала захвата изображения. С помощью IMAQdx Get Image происходит захват первого кадра, который в дальнейшем используется детектирования перемещения в первый момент времени. Для улучшения производительности за счёт обработки изображения более сжатого формата используем IMAQ Cast Image для конвертации полученного изображения в монохромный тип изображения (яркость пикселя определяется беззнаковым 8-битным числом). Так же для улучшения производительности используем IMAQ Resample для уменьшения разрешения изображения до 120x160 пикселей, что достаточно для детектирования скорости перемещения

всего изображения. IMAQ Create используется для создания ячейки памяти, в которой хранится изображение. Входным параметром подпрограммы служит строковое значение – имя ячейки. Выходным параметром подпрограммы служит ссылка на эту ячейку памяти с изображением.

Для детектирования перемещения с помощью алгоритма подпрограммы IMAQ Optical Flow (HS) необходимо использовать два изображения — предыдущий кадр (Prev Frame) и текущий кадр (Current Frame). Подпрограммы IMAQ Optical Flow (HS/LK) выдаёт два изображения Velocity Component 1 и Velocity Component 2. Velocity Component 1 и Velocity Component 2 содержат в себе монохромное изображение, яркость каждого пикселя которого определяет детектированную скорость перемещения изображения, соответственно, по координате X и по координате Y в области этого пикселя.

Для хранения координат положения роботизированной платформы используем сдвиговый регистр, на вход которого подан кластер, который содержит координаты x, y в миллиметрах и направление роботизированной платформы в градусах.

Для хранения пути, по которому должен двигаться робот используется сдвиговый регистр, на вход которого пода массив точек этого пути. Каждая точка — кластер, содержащий в себе x и y координаты точки.

Для графического представления положения роботизированной платформы и пути используется «XY Graph».

IMAQ Copy используется для копирования изображения из памяти текущего кадра в память предыдущего кадра для использования этого изображения в следующей итерации.

Для того, чтобы предотвратить перемещение, в случае, когда изображение еще не поступило, необходимо в конце каждой итерации цикла сначала включать движение на какой-то промежуток времени, а потом отключать движение. Для того, чтобы последовательность действий была строго задана, использована структура «Flat Sequence Structure». В первом «кадре» структуры использована пользовательская подпрограмма «joy_move.vi», которая преобразует полученные данные управления в понятные данные для

роботизированной платформы. Входными данными является кластер булевых значений, получаемых от «auto_moving.vi», булевы значения состояния кнопок управления перемещением на лицевой панели, данные, получаемые от джойстика с помощью подпрограммы «Acquire Input Data» и число от «Ring Properties», которое выбирает источник управления роботизированной платформой (программно — 0, кнопками лицевой панели — 1, джойстиком — 2). Данные в цикл работы с сетью передаются с помощью локальных переменных.

Все дальнейшие действия выполняются вне структуры «Flat Sequence Structure», так как синхронизация потоков далее не требуется.

«Initialize Joystick» инициализирует работу с джойстиком, который используется для управления роботизированной платформой в ручном режиме.

Булева переменная Arm On/Off содержит в себе начальное состояние включения сервоприводов манипулятора.

Для инициализации связи по протоколу TCP/IP использует подпрограмма TCP Open Connection, входными параметрами которого служат IP-адрес роботизированной платформы — 192.168.33.2, уставка таймаута (3000 мс) и порт 2055.

4.1 Описание цикла обработки изображения и логики управления роботизированной платформой

Для того, чтобы избежать рывков изображения из-за того, что при обработке изображения и вывода его на экран потоки данных не синхронизированы используем структуру «Flat Sequence Structure». Такая структура обеспечивает синхронизацию за счёт того, что каждый последующий «кадр» этой структуры не начнётся выполняться, пока не поступят данные на все входы следующего кадра. В первом кадре этой структуры происходит захват изображения с помощью IMAQdx Get Image. Для улучшения производительности за счёт обработки изображения более сжатого формата

используем IMAQ Cast Image для конвертации полученного изображения в монохромный тип изображения (яркость пикселя определяется беззнаковым 8-битным числом). Так же для улучшения производительности используем IMAQ Resample для уменьшения разрешения изображения до 120x160 пикселей, что достаточно для детектирования скорости перемещения всего изображения.

Структура «Case Structure» предназначена для возможности выбора алгоритма обработки изображения прямо при выполнении программы. Библиотека IMAQ предоставляет два алгоритма для вычисления перемещения изображения.

Первая подпрограмма IMAQ Optical Flow (LK) реализует алгоритм Лукаса и Канаде (Lucas and Kanade algorithm). Такой алгоритм является сравнительно быстрым, но менее точным.

Вторая подпрограмма IMAQ Optical Flow (HS) реализует алгоритм алгоритма Хорна и Шунка (Horn and Schunck algorithm). Такой алгоритм является более точным, но также более требовательным к вычислительным мощностям компьютера.

Входными данными обоих подпрограмм являются ссылки на изображения предыдущего кадра, текущего кадра, и изображения компонентов перемещения Velocity Component 1 и Velocity Component 2

Выходными данными обоих подпрограмм являются ссылки на изображения Velocity Component 1 и Velocity Component 2.

Во втором «кадре» структуры «Flat Sequence Structure» реализовано получения численного значения скорости перемещения с помощью пользовательской подпрограммы «flow_speed.vi».

4.2 Описание пользовательской подпрограммы «flow_speed.vi»

Входными данными подпрограммы являются:

а) «Velocity Component 1 In» – ссылка на изображение Velocity Component 1. Используется для определения горизонтального потока изображения;

б) «Velocity Component 2 In» – ссылка на изображение Velocity Component 2. Используется для определения вертикального потока изображения;

в) «Pos in» – кластер, содержащий в себе положение роботизированной платформы в предыдущий момент времени. Определяется тремя значениями типа Double – координата *x*, координата *y* и угол направления против часовой стрелки относительно координатной оси OX;

г) «Speed Mul» – значение типа Double – коэффициент, определяющий соотношение реальной скорости перемещения вперёд роботизированной платформы и средней скорости оптического потока вертикально;

д) «Deg Mul» значение типа Double – коэффициент, определяющий соотношение реальной угловой скорости поворота роботизированной платформы и средней скорости оптического потока горизонтально;

е) «Program Control In» – кластер, содержащий четыре булевых значения. Каждое булево значение означает движение или вращение в одну из сторон – движение вперёд, движение назад, поворот налево и поворот направо.

Выходными данными подпрограммы являются:

а) «Pos out» – кластер, содержащий в себе положение роботизированной платформы в следующий момент времени. Определяется тремя значениями типа Double – координата *x*, координата *y* и угол направления против часовой стрелки относительно координатной оси OX;

б) «Graph Pos out cluster» – графическое представление положения роботизированной платформы на графике «Graph XY»;

в) «Horizontal Flow» – средняя скорость горизонтального оптического потока;

г) «Vertical Flow» – средняя скорость вертикального оптического потока.

Для определения средней скорости оптического потока необходимо получить среднее значение яркости пикселя изображения «Velocity Component 1 In» – для горизонтального потока и «Velocity Component 2 In» – для вертикального потока. Определение средней яркости пикселя происходит с помощью подпрограммы «IMAQ Histograph VI» из библиотеки «NI Vision Development Module». Входным параметром функции является ссылка на изображение «Image», выходным – «Mean Value» – значение типа Single – средняя яркость пикселя изображения.

Значение горизонтального оптического потока умножается на коэффициент «Deg Mul», таким образом получаем величину угла в градусах, на которую повернулась роботизированная платформа за время, прошедшее с предыдущего цикла выполнения программы. Это значение складывается со значением угла направления роботизированной платформы из кластера «Pos in». Полученное значение является новым значением угла направления роботизированной платформы.

Значение вертикального оптического потока умножается на коэффициент «Speed Mul», таким образом получаем расстояние в мм, которое прошла роботизированная платформа за время, прошедшее с предыдущего цикла выполнения программы. Полученное значение используется для расчёта нового положения роботизированной платформы по координатам x и y по формулам (1) и (2).

$$x = x_p + \upsilon \cdot \cos(\alpha), \tag{1}$$

где x_p – координата оси OX положения роботизированной платформы в предыдущий момент времени;

α – угол против часовой стрелки, измеряемый в градусах между осью OX и роботизированной платформы.

$$y = y_p + \upsilon \cdot \cos(\alpha), \quad (2)$$

где y_p – координата оси OY положения роботизированной платформы в предыдущий момент времени.

Полученные координаты x, y и угол α объединяются в кластер «Pos out» и являются выходными данными, отражающие новое положение роботизированной платформы.

Для предотвращения возникающих помех, связанных с появлением движущихся объектов в поле зрения или из-за возникающих помех в момент, когда роботизированная платформа не движется происходит отмена обновления положения роботизированной платформы.

В момент, когда роботизированная платформа движется только вперёд или назад, обновление угла направления роботизированной платформы не происходит, так как заведомо известно, что при таком движении направление роботизированной платформы измениться не должно.

Аналогично, в момент, когда роботизированная платформа производит поворот, обновление положения по координатам x и y не происходит, так ка заведомо известно, что при таком движении положение по координатам x и y роботизированной платформы измениться не должно, так как разворот происходит вокруг своей оси.

«Graph Pos out cluster» состоит из массива размером две ячейки. Каждая из ячеек – кластер, состоящий из двух чисел типа Double, отражающие положение роботизированной платформы на координатной плоскости «Graph XY». Первая точка совпадает с положением роботизированной платформой, вторая – на 200 мм дальше вдоль направления роботизированной платформы.

Код подпрограммы flow_speed.vi изображён на рисунке 6.

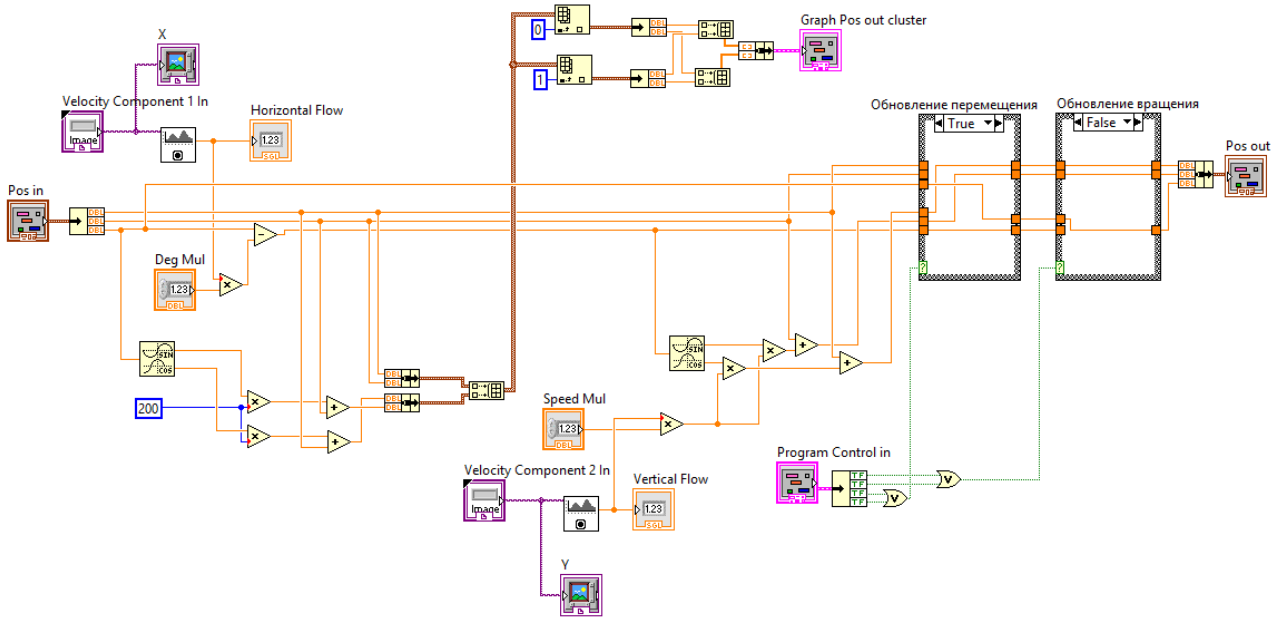


Рисунок 6 — Код подпрограммы flow_speed.vi

4.3 Описание пользовательской подпрограммы «check_arrived_point.vi»

Пользовательская подпрограммы «check_arrived_point.vi» используется для проверки, достигла ли роботизированная платформа первой точи пути, и если да, то происходит удаление из массива пути этой точки. По достижении этой точки, роботизированная платформа начинает двигаться к следующей точке.

Входными этой подпрограммы являются:

а) «Pos in» – кластер двух массивов – список координат *x* и список координат *y*, представляющее графическое представление позиции роботизированной платформы на графике «Graph XY». Первый индекс массива – координаты *x* и *y* роботизированной платформы на координатной плоскости;

б) «Path» – кластер двух массивов – список координат *x* и список координат *y*, отражающее путь в предыдущий момент времени, который необходимо пройти роботизированной платформе;

в) «Dist» – тип Double – манхэттенское расстояние от роботизированной платформы до точки, при которой можно считать, то эта точка достигнута роботизированной платформой. При этом точка удаляется из массива пути.

Выходными данными является «Cluster out» – кластер двух массивов – список координат *x* и список координат *y*, отражающее путь в следующий момент времени.

Точка пути считается достигнутой, при условии, что первая точка массива «Pos» и первая точка массива «Path» отстоят не дальше друг от друга больше, чем манхэттенское расстояние, указанное в параметре «Dist».

Код подпрограммы изображён на рисунке 7.

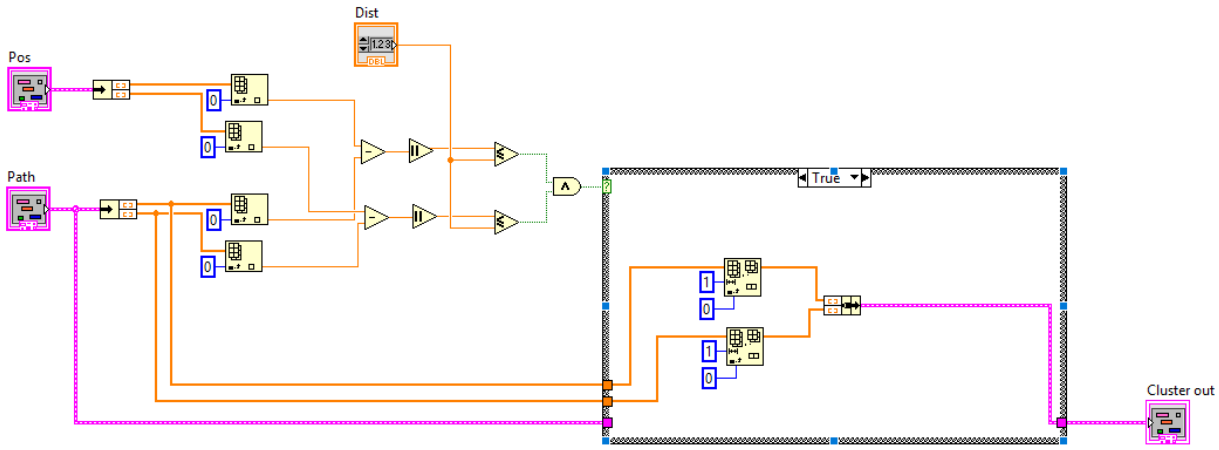


Рисунок 7 — Код подпрограммы «check_arrived_point.vi»

4.4 Описание пользовательской подпрограммы «auto_moving.vi»

С помощью пользовательской подпрограммы auto_moving.vi осуществляется управление перемещением на основе координат роботизированной платформы и пути, по которой должна она следовать. Входными данными являются:

а) «Pos in» – кластер из трёх чисел типа Double, содержащий координаты роботизированной платформы – координаты *x*, *y* и угол против часовой стрелки *α* между осью координат OX и направлением роботизированной платформы;

б) «Путь in» – массив точек пути.

Выходными данными является кластер, состоящий из четырёх булевых переменных — эквивалентных четырём кнопкам джойстика для управления в ручном режиме. Первая переменная — поворот налево, вторая — направо, третья — перемещение вперёд, четвёртая — назад.

Координаты точки, до которой роботизированная платформа должна идти определены нулевым индексом массивов кластера «Путь in». Для определения угла направления от точки расположения роботизированной платформы до точки, до которой роботизированная платформа должна двигаться используется пользовательская подпрограмма «coord_to_angle.vi». Входные данные функции – координаты *x* и *y* точки, до которой нужно двигаться и точки, расположения роботизированной платформы. Выходные данные – число типа Double – угол в градусах против часовой стрелки между осью OX координатной плоскости и углом направления от точки расположения роботизированной платформы до точки, в которую роботизированная платформа должна двигаться. Для проверки, направления роботизированной платформы и коррекции направления используется пользовательская функция «get_rotation_direction.vi». Функция принимает на вход два числа типа Double – текущее направление в градусах роботизированной платформы и направление, по которому должна двигаться роботизированная платформа, чтобы достичь заданной точки. Так как функция использует углы в градусах, а угол направления роботизированной платформы определён в радианах, то для перевода величины угла из радиан в градусы используется пользовательская функция «rad_to_deg.vi». Выходными данными функции «get_rotation_direction.vi» являются:

а) «Direction (Left = -1, Right = 1, Error = 0)» – число типа I32 (знаковый целочисленный), которое может принимать значения «-1», «0», и «1». Если выходным числом является «-1», то для того, чтобы роботизированной

платформы достичь заданного направления, необходимо повернуться влево. Аналогично, при выходном числе «1» роботизированной платформе необходимо повернуться вправо. Значение «0» сигнализирует об ошибке выполнении функции;

б) «Absolute Angle Difference (degrees)» – число типа Double – наименьший угол между двумя направлениями.

Движение роботизированной платформы вперёд происходит при условии, что угол между направлением роботизированной платформой и направлением движения до заданной точки менее 10°. В этом случае кластер «Program Control out» принимает значение, соответствующее движению вперёд роботизированной платформы.

В случае, когда угол между направлением роботизированной платформой и направлением движения до заданной точки равно или более 10°, то значение кластера принимает «Program Control out» соответствующее значение, при котором роботизированная платформа корректирует своё направление вправо или влево.

При возникновении ошибки в функции «get_rotation_direction.vi» роботизированная платформа останавливает своё движение.

Код подпрограммы изображён на рисунке 8.

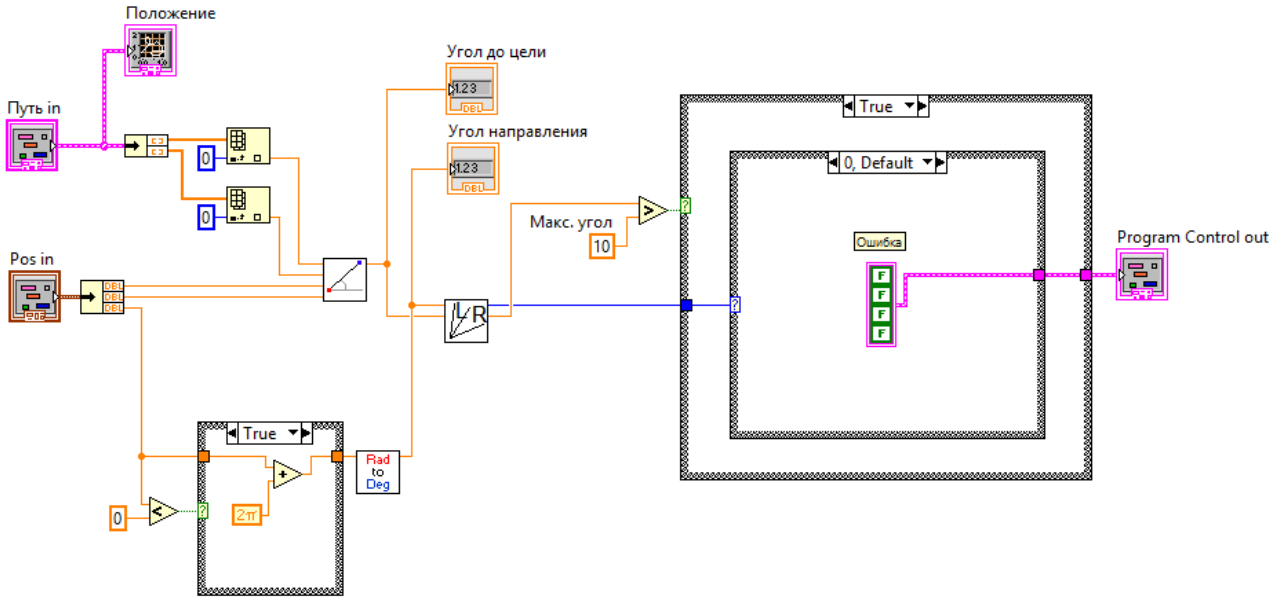


Рисунок 8 — Код подпрограммы «auto_moving.vi»

4.5 Описание пользовательской подпрограммы «get_rotation_direction.vi»

Для определения направления вращения и угла между векторами создадим пользовательскую функцию «get_rotation_direction.vi». На вход функции подаются два значения типа Double:

а) «Angle Source (degrees)» – угол направления робртизированной платформы в градусах (вектор $\vec{r}$);

б) «Angle Cell (degrees)» – угол направления вектора $\vec{r}'$.

Выходными значениями функции являются:

а) «Absolute Angle Difference (degrees)» – значение типа Double – угол между двумя векторами в градусах, в диапазоне от 0° до 179°;

б) «Direction (Left = -1, Right = 1, Error = 0)» – значение типа I32 (знаковое целочисленное). Принимает значения 0, -1 или 1, в зависимости от направления вращения. «-1» – соответствует вращению против часовой стрелки, значение

«1» – соответствует направлению вращения по часовой стрелки, значение «0» – обозначает ошибку при выполении функции.

Для того, чтобы быть в дальнейшем уверенным, что угол лежит в пределах 0° до 359°, необходимо входные значения поделить на 360.

Для определения угла между векторами используем стандартную функцию «Absolute Angle Difference VI».

Для определения направления вращения, необходимо определить, в каком квадранте находятся углы. Для этого используем пользовательскую функцию «get_quadrant.vi». Входным значением этой функции является велечина угла, выходным значением является номер квадранта – 1, 2, 3 или 4 соответственно. В некоторых случаях, направление вращения можно пределить по номерам квадрантов входных углов. В таблице 1 определены случаи, при которых определены направления вращения по номерам квадрантов.

Таблица 1 — Направления вращения, соотвтетствующие номерам квадрантов углов

Квадрант угла α	Квадрант угла α’	Направление вращения
1	1	—
1	2	Против часовой стрелки
1	3	—
1	4	По часовой стрелке
2	1	По часовой стрелке
2	2	—
2	3	Против часовой стрелки
2	4	—
3	1	—
3	2	По часовой стрелке
3	3	—
3	4	Против часовой стрелки
4	1	Против часовой стрелки
4	2	—
4	3	По часовой стрелке
4	4	—

Можно заметить, что в 8 случаях из 16 направление вращения невозможно определить по номерам квадрантов входных углов. Это случаи, когда номера квадрантов обоих углов совпадают, или их квадранты направлены в противоположные стороны, то есть 1 и 3, 2 и 4, 3 и 1, 4 и 2.

В случае, когда оба угла находятся в одном квадранте, то определить направление вращения можно путём проверки, который из углов имеет большее значение. Если угол вектора $\vec{r}$ окажется больше угла вектора $\vec{r}'$, то вращение должно быть по часовой стрелки, в противном случае – против часовой стрелки.

В случае, когда углы находятся в противоположных квадрантах, необходимо из наибольшего угла вычесть 180. Таким образом получится, что оба угла находятся в одном квадранте. Далее, необходимо проделать проверку, который из углов имеет большее значение. Если угол вектора $\vec{r}$ окажется больше угла вектора $\vec{r}'$, то вращение должно быть по часовой стрелки, в противном случае – против часовой стрелки.

Так как структура «Case Structure» не может сравнивать пау чисел с дрйгой парой числе, то вместо это используем сравнения строк размером в два символа – цифра номера квадранта первого угла и цифра номера квадранта второго угла.

Исходный код пользовательской функции «get_rotation_direction.vi» изображён на рисунке 9.

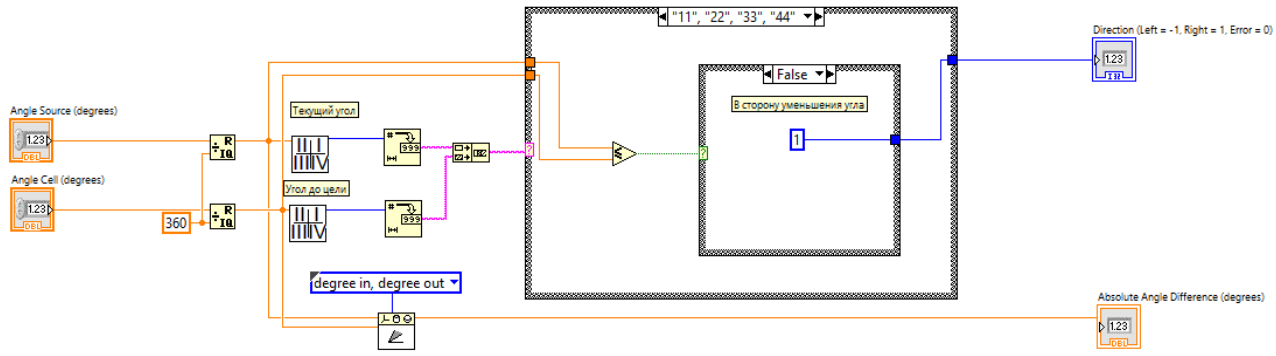


Рисунок 9 — Исходный код пользовательской функции «get_rotation_direction.vi»

4.6 Описание пользовательской подпрограммы «get_quadrant.vi»

Пользовательская функция «get_quadrant.vi» используется для определения номера квадранта угла. Функция «get_quadrant.vi» имеет один вход типа Double – входной угол, и один выход типа I32 (целочисленный знаковый) – номер квадранта 1, 2, 3 или 4. Выходное значение «0» соответсвует ошибке при выполнении функции.

Для определения номера квадранта достаточно знать, в каком диапазоне находится значение угла. В таблице 2 диапазоны значений углов, соотвтетствующие номерам квадрантов.

Таблица 2 — Диапазоны значения угла, соотвтетсвующие номеру квадранта

Минимальное значение (включая)	Максимальное значение (не включая)	Номер квадранта
0	90	1
90	180	2
180	270	3
270	360	4

Для того, чтобы быть в дальнейшем уверенным, что угол лежит в пределах 0° до 359°, необходимо входное значение поделить на 360. Для того, чтобы отловить случаи, когда ни одна проверка не прошла, или прошли две и более проверки, сделаем проверку для отлова таких случаев. При возникновении таких случаев функция выдаёт значение «0».

Исходный код пользовательской функции «get_quadrant.vi» изображён на рисунке 10.

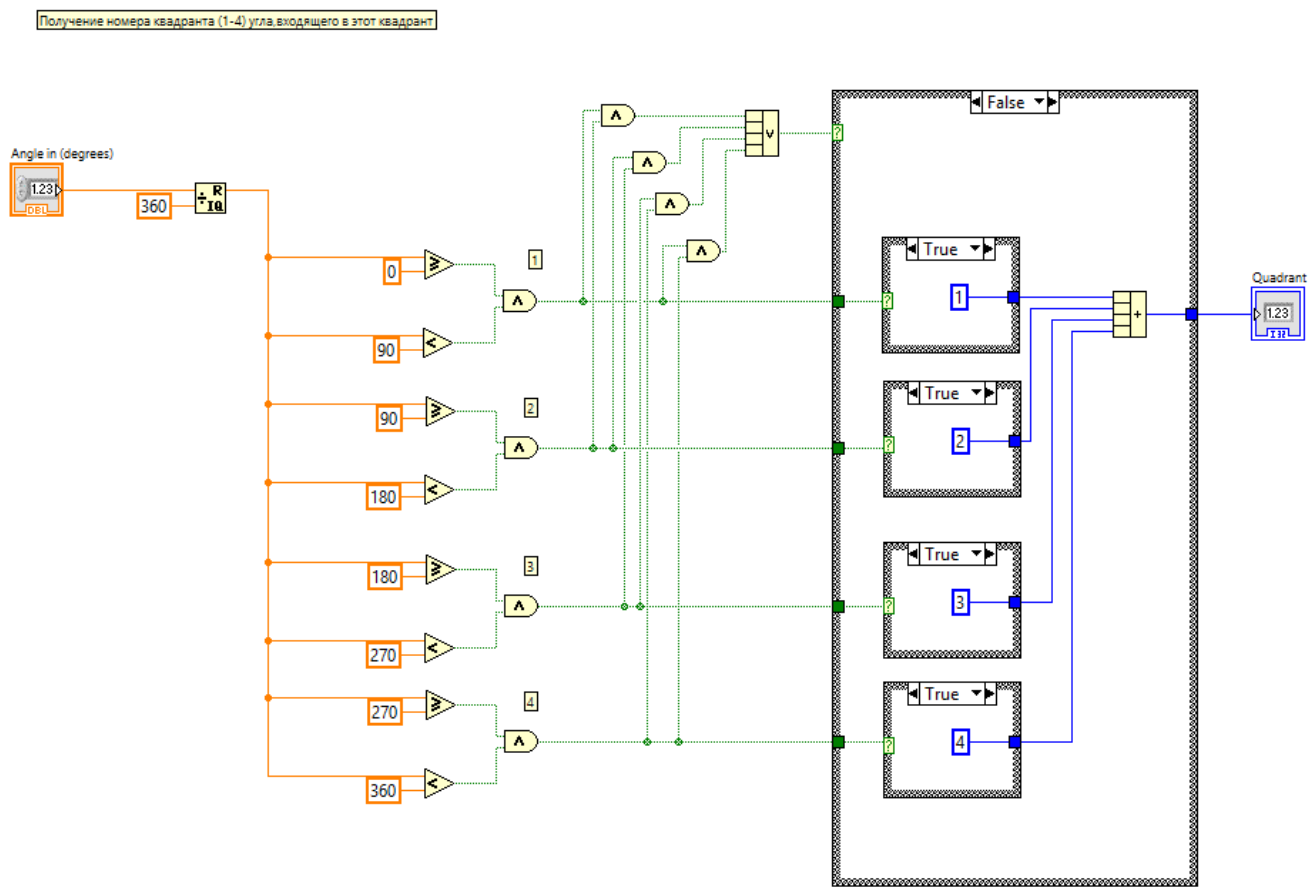


Рисунок 10 — Исходный код пользовательской функции «get_quadrant.vi»

4.7 Описание пользовательской подпрограммы «coord_to_angle.vi»

Для определения угла между отрезком, заданного двумя точками и осью OX оси координат создадим пользовательскую функцию «coord_to_angle.vi». На вход функции подаются четыре значения типа Double «X1», «Y1», «X2» и «Y2», соответсвующие координатам точки отрезка.

Выходными значениями функции является значение типа Double «Angle out» – значение угла в градусах против часовой стрелки между осью координат OX и отрезка заданного двумя точками.

Для определения угла отрезка, заданного двумя точками и осью OX оси координат, необходимо использовать формулу косинуса угла между двумя векторами заданных координатами (1).

$$\cos\alpha = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \cdot \sqrt{x_2^2 + y_2^2}} \quad (1)$$

Так как единичный вектор $\{x_2; y_2\}$ вдоль оси OX имеет координаты {1; 0}, тогда формула (1) получит вид (2).

$$\cos\alpha = \frac{x_1}{\sqrt{x_1^2 + y_1^2}} \quad (2)$$

Так как отрезок задан двумя точками $(x_1; y_1)$ и $(x_2; y_2)$, то координаты вектора, лежащего вдоль этого отрезка имеют значение $\{x_1 - x_2; y_1 - y_2\}$. Тогда формула (2) прмет вид (3).

$$\cos\alpha = \frac{x_1 - x_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \quad (3)$$

Для того, чтобы получить значение угла, необходимо полученное значение вычислить через функцию arccos, которая выдаёт значение в радианах. Для перевода в градусы полученное значение необходимо умножить на 180 и разделить на π. Тогда формула (3) прмет вид (4).

$$\alpha = \frac{180}{\pi} \cdot \arccos\frac{x_1 - x_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \quad (4)$$

В результате, получим значение наименьшего угла между осью OX и отрезком, заданного двумя точками. Так как нам необходимо значение угла, только против часовой стрелки, то необходимо сделать несколько проверок. Известно, что угол вектора, лежащего вдоль оси OX всегда находится в первом квадранте. Чтобы найти номер квадранта угла вектора, лежащего вдоль отрезка,

достаточно проверить знак координаты *y*. Если знак положительный, то угол определён против часовой стрелки, если знак отрицательный, то угол определён по часовой стрелке. Чтобы найти угол против часовой стрелки, необходимо из 360 вычесть значение угла, найденного по часовой стрелке.

Исходный код пользовательской функции «coord_to_angle.vi» изображён на рисунке 11.

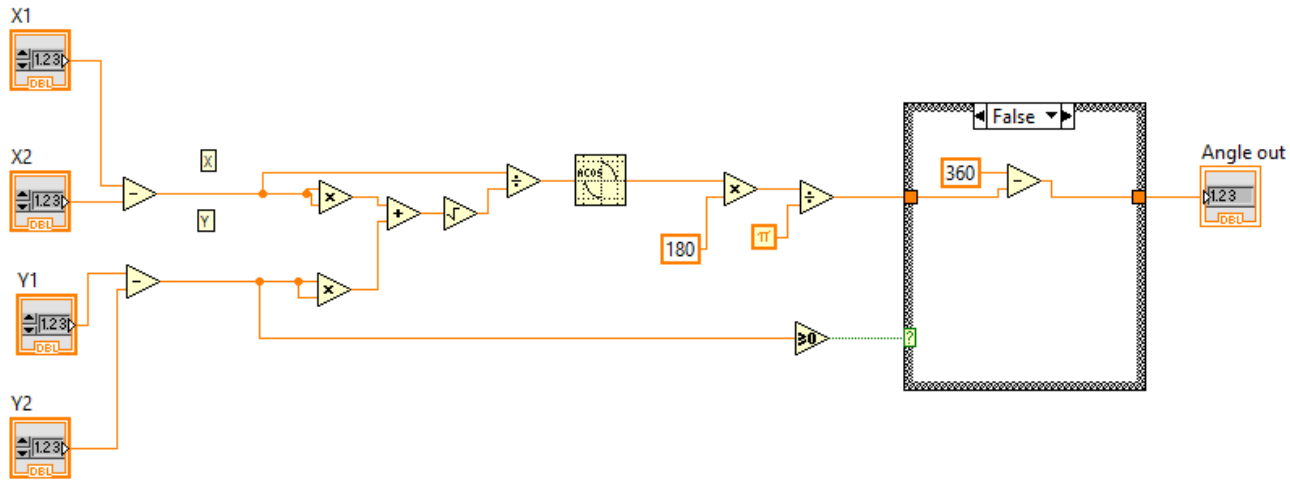


Рисунок 11 — Исходный код пользовательской функции «coord_to_angle.vi»

Код подпрограммы «joy_move.vi» изображён на рисунке 12.

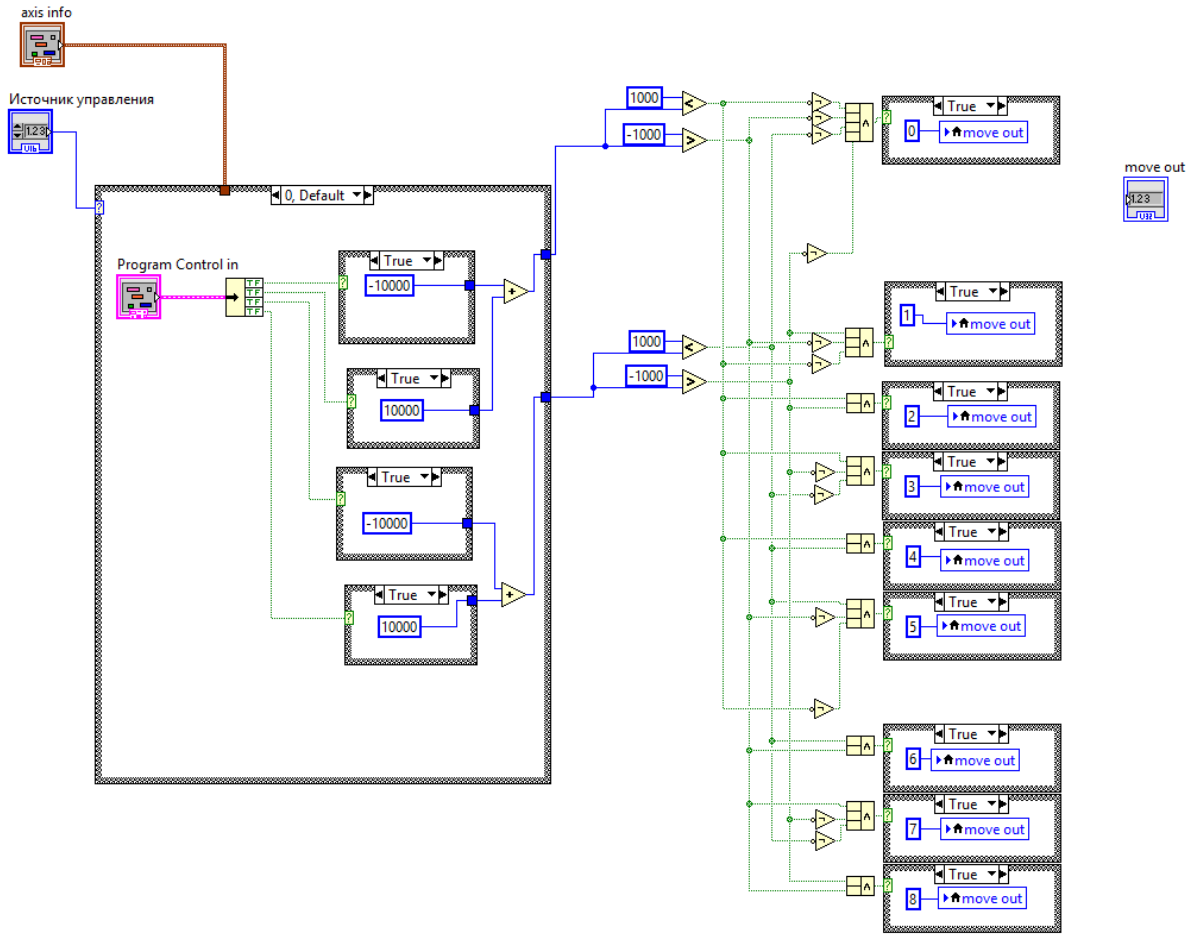


Рисунок 12 — Код подпрограммы joy_move.vi

Для управления сервоприводами манипулятора используется структура «Case Structure», которая в зависимости от включения тумблёра на лицевой панели «Манипулятор Вкл./Выкл.» включает и выключает управление сервоприводами. Угол поворота сервоприводов зависит от положения регуляторов на лицевой панели. Данные в цикл работы с сетью так же передаются с помощью локальных переменных.

Завершение цикла происходит по получению соответствующего значению локальной переменной при завершении программы. По завершению программы происходит отчистка памяти изображений, отключение захвата камеры.

Код цикла обработки изображения и логики управления роботизированной платформой изображён на рисунке 13.

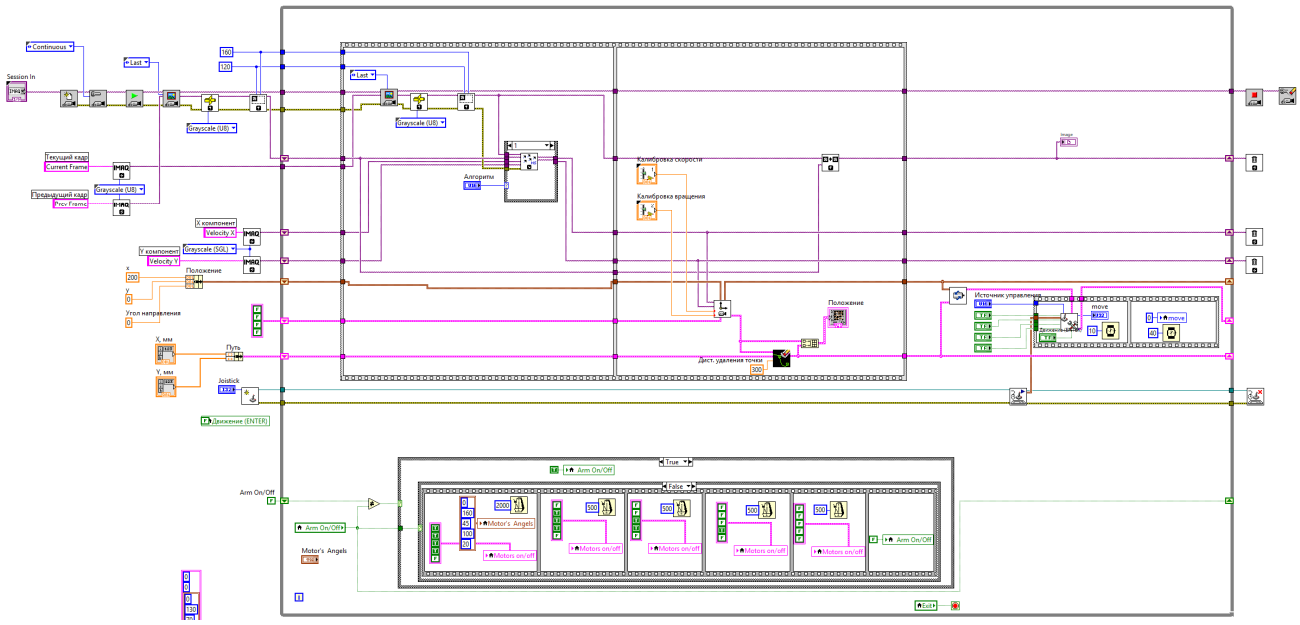


Рисунок 13 — Код цикла обработки изображения и логики управления роботизированной платформой

4.8 Описание цикла работы с передачей данных

Передача данных осуществляется по протоколу TCP/IP. Конвертация данных в формат для передачи данных осуществляется с помощью подпрограммы Flatten To String. С помощью подпрограммы TCP Write происходит запись данных в сетевое соединение. Чтение данных осуществляется с помощью подпрограммы TCP Read. С помощью подпрограммы Unflatten From String происходит конвертация полученных данных дальнейшей работы с ними. Эти данные передаются в цикл обработки изображения и логики управления роботизированной платформой с помощью локальных переменных. По завершению записи и чтения происходит задержка в 100 мс. Выход из цикла происходит по нажатию кнопки Exit на лицевой панели. По завершению цикла происходит разрыв соединения TCP. Код цикла работы с передачей данных изображён на рисунке 13.

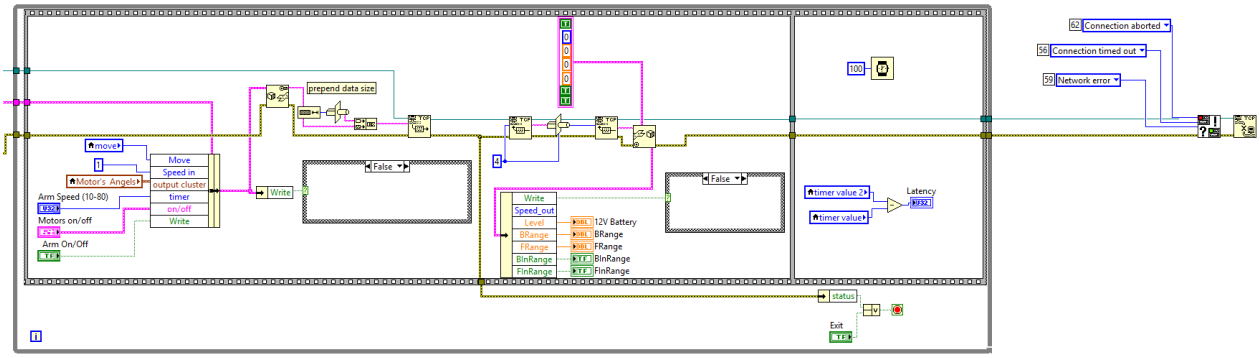


Рисунок 13 — Код цикла работы с передачей данных

5 Описание лицевой панели программы

На лицевой панели представлены все элементы управления. При установлении связи с роботизированной платформой на лицевой панели показывается изображение, захваченное с камеры. На операторной панели присутствуют органы управления манипулятором и шасси. На лицевой панели можно так же выбрать способ управления платформой (программно, с помощью джойстика, с помощью кнопок лицевой панели). На лицевой панели отображается положение роботизированной платформы, а также пути, по которому необходимо следовать. Так же отображается уровень заряда аккумуляторной батареи, состояние связи, данных, полученных с ИК-датчиков приближения к препятствиям. На лицевой панели так же можно отрегулировать калибровку измерения скорости перемещения и вращения, а также выбрать алгоритм детектирования скорости по изображению с камеры.

Лицевая панель показана на рисунке 14.

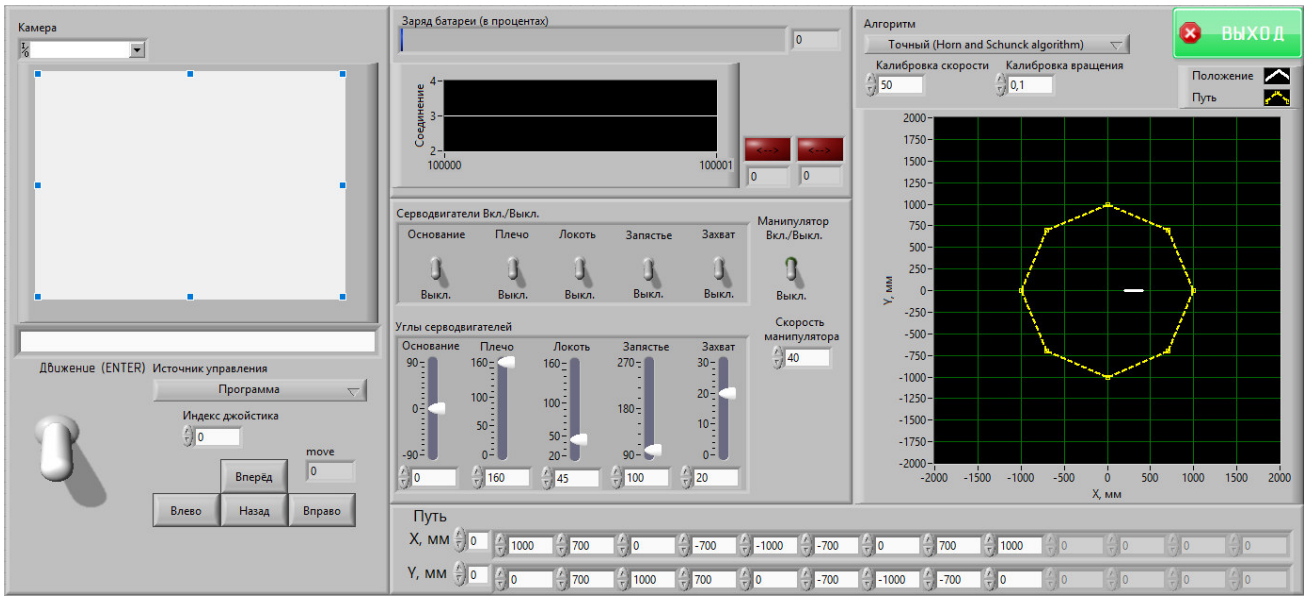


Рисунок 14 — Лицевая панель программы

6 Экспериментальное подтверждение работоспособности алгоритма

Мобильная роботизированная платформа перемещается для решения тех или иных задач, получает данные с внешних датчиков, и должен постоянно обрабатывать информацию, чтобы управлять своим движением. Обработка информации и управление движением осуществляется с помощью программы, выполняемой на удалённом компьютере. Все эти процессы происходят непрерывно и тесно взаимосвязаны друг с другом.

Если предположить, что робот ограничивается перемещением на плоскости, то его местоположение может быть определено тремя параметрами:

а) координата X относительно начального положения платформы;

б) координата Y относительно начального положения платформы;

в) угол направления между осью OX координатной плоскости и углом текущего направления роботизированной платформы.

Оценка движения роботизированной платформы осуществляется путем определения скорости перемещения камеры на основе видеоданных по одному из методов — по алгоритму Лукаса и Канаде или по алгоритму Хорна и Шунка. Движение роботизированной платформы осуществляется по траектории, записанного в памяти удалённого компьютера. Траектория движения описана точками на координатной плоскости. На рисунке 15 изображена индикация траектории движения и положения робота на лицевой панели программы, выполяемой на удалённом компьютере.

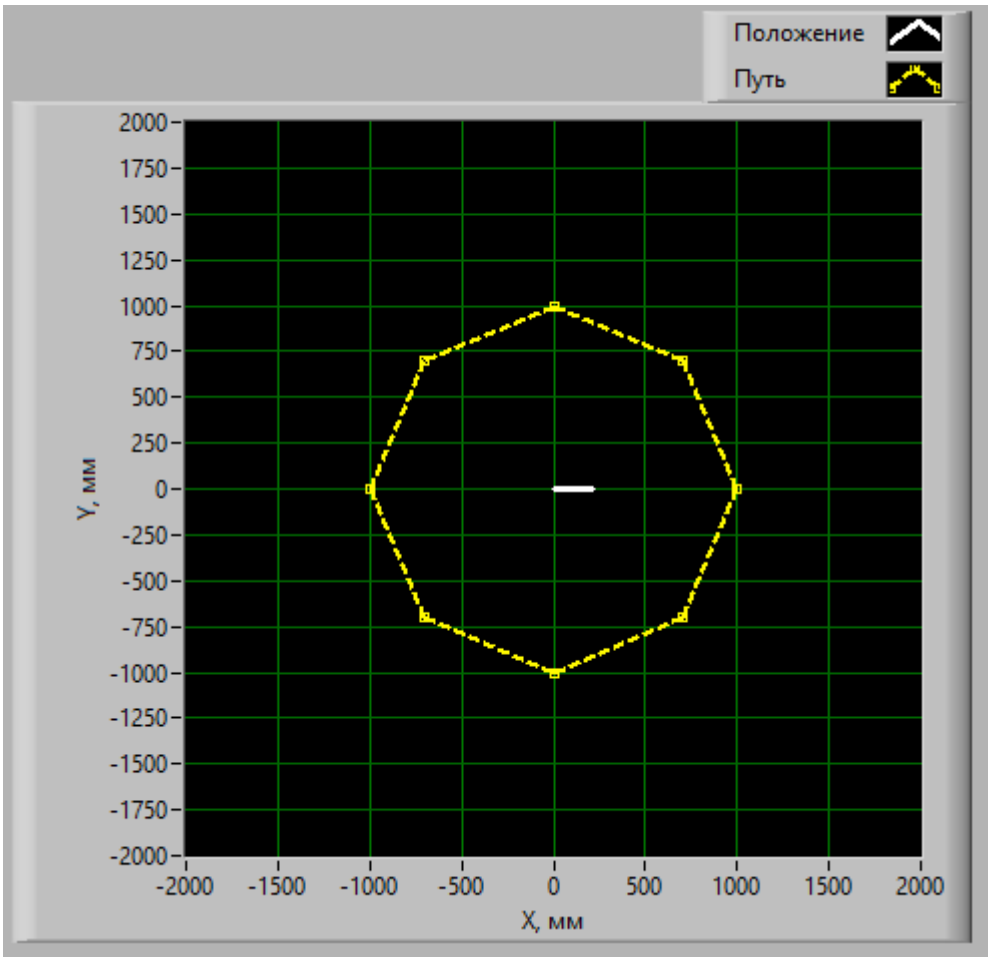


Рисунок 15 — Индикация траектории движения и положения робота на лицевой панели

Для того, чтобы реально пройденный путь движения был наиболее приближенным к заданной траектории движения, необходимо провести калибровку.

Калибровка включает экспериментальное перемещение робота и сравнение фактического значения пройденного расстояния, угловой скорости, со значениями, полученным в результате теоретической оценки. Отношение фактического к теоретическому значениям и является коэффициентом калибровки. Для калибровки скорости прямолинейного движения и угловой скорости используются два коэффициента калибровки. На рисунке 16 изображены формы для записи значений коэффициентов на лицевой панели.

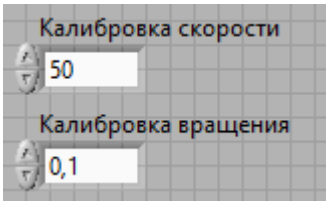


Рисунок 16 — Формы для записи значений коэффициентов на лицевой панели

Если при повторных испытаниях будет наблюдаться расхождения в полученных значениях, мы можем повысить точность путем изменения значения коэффициентов на лицевой панели, а затем повторить процесс.

Было установлено, что наибольшая точность перемещения по заданной траектории достигается при коэффициентах калибровки 50 — для калибровки прямолинейного движения и 0,1 — для калибровки угловой скорости при поворотах.

Было экспериментально подтверждена работоспособность используемой программы для движения по заданному маршруту. Однако, при движении выявлены погрешности, которые могут сильно сказываться на точности позиционирования.

7 Выявление и исключение погрешностей позиционирования

Наибольшие погрешности в определении текущего положения вносятся на стадии выполнения алгоритма определения скорости перемещения камеры на основе видеоданных. Вид таких погрешностей имеет мультипликативный характер, так как с получением каждого нового кадра происходит вычисление позиции на основе предыдущих данных.

Причин возникновения таких погрешностей может быть несколько:

а) тряска камеры во время движения;

б) плохое качество изображения (зернистость, несфокусированность, недостаточное разрешение изображения и др.);

в) появление движущихся предметов в кадре относительно пола;

г) недостаточное освещение, или чрезмерно сильное освещение помещения, в результате чего, изображение может быть расплывчатым или однородным;

д) однородная поверхность пола, в результате чего программа определения скорости перемещения камеры не сможет найти «точки опоры», относительно которых определяется передвижение;

е) загруженность сети передачи данных, в результате чего видеопоток может поступать с задержками;

ж) низкая производительность удалённого компьютера, которая не позволяет своевременно обработать полученные данные;

з) неправильное расположение камеры на автоматизированной платформе, установлен неправильный угол направления камеры;

и) несовершенство используемых алгоритмов.

7.1 Погрешности, вносимые несовершенством алгоритмов

Было установлено, что алгоритм вычисления Хорна и Шунка вносит меньше погрешностей, чем алгоритм Лукаса и Канаде. Однако алгоритм Хорна и Шунка требует больших вычислительных мощностей, поэтому пользователю предоставлен выбор, какой алгоритм будет использоваться при определении скорости перемещения камеры. На рисунке 17 изображено выпадающее меню на лицевой панели для выбора алгоритма определения скорости перемещения камеры.

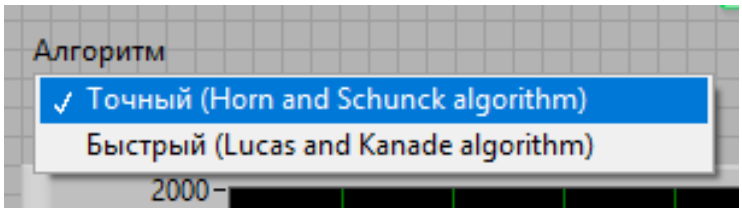


Рисунок 17 — Выбор алгоритма определения скорости перемещения камеры с помощью выпадающего списка на лицевой панели

Основной недостаток реализации используемых алгоритмов в LabView — неточный поиск точек опоры на изображении, относительно которых осуществляется вычисление скорости перемещения картинки. Обнаруженные опорные точки могут «теряться» программой или могут быть приняты за соседние опорные точки.

7.2 Погрешности, возникающие из-за движущихся предметов

Было обнаружено, что при появлении движущихся объектов относительно поверхности пола (люди, домашние животные, открывающиеся двери и т.д.) в области захвата изображения появляются дополнительные случайные погрешности. Это происходит из-за того, что алгоритмы воспринимают движущиеся области изображения как движения в результате передвижения камеры в пространстве. Для уменьшения вероятности появления таких случаев камера должна быть наклонена к полу примерно на 45° – 60°. Это существенно сократит область, в которой появление движущихся объектов вносит дополнительные погрешности.

7.3 Погрешности, возникающие из-за непостоянной скорости получения данных

Видеопоток может быть не всегда обработан с одинаковой скоростью. Так же, получение изображений (кадров) может происходить с разной скоростью в разные моменты времени, например, из-за загруженности сети. Более того, поступление изображений может прекратиться на какое-то время, или даже произойти обрыв связи. В результате того, что процесс получения информации об окружающем мире может происходить с разной скоростью, это увеличивает погрешность позиционирования, так как роботизированная платформа может не успевать среагировать на внешние изменения (изменение положения в пространстве) из-за отсутствия актуальных данных и может проехать большее расстояние, чем необходимо для достижения нужной точки. Для исключения таких случаев в программу внесена защита, которая останавливает движение роботизированной платформы, если с прошлого поступления данных прошло слишком много времени (более 50 мс). Роботизированная платформа возобновляет движение после того, как данные с камеры будут вновь получены и обработаны.

7.4 Погрешности, возникающие из-за несовершенства качества изображения

Из-за зернистости изображения, два последовательных кадра могут сильно отличаться друг от друга, даже если камера неподвижна. Эти изменения могут быть почти незаметны человеческому глазу, однако, программа, основанная на алгоритмах Лукаса и Канаде или Хорна и Шунка может принять их за результат хаотичного движения камеры в пространстве. Для уменьшения такой погрешности используется фильтр изображения подавления зернистости. Для еще большего эффекта детектирование перемещения камеры в пространстве отключается в тот момент, когда роботизированная платформа явно находится в покое (то есть, когда роботизированной платформе не поступает сигнал для начала движения). При движении прямолинейно, отключается детектирование угловой скорости роботизированной платформы, а при вращении отключается детектирование скорости прямолинейного движения.

ЗАКЛЮЧЕНИЕ

Основной целью работы была разработка программного обеспечения роботизированной платформы, предназначенной для обследования помещений.

Для достижения этой цели перед нами был поставлен ряд задач, касающихся изучения существующих методов определения оптического потока и позиционирования.

Существующие методы определения оптического потока видеоизображения были изучены, и был реализован соответствующий алгоритм, позволяющий решить задачу позиционирования. Были изучены методы распознавания перемещения камеры в пространстве на основе полученных видеоданных, что также было реализовано в работе в виде приложения в графической среде программирования LabVIEW.

Были определены оптимальные условия освещения и прочих параметров помещения, при которых достигается наиболее точное позиционирование роботизированной платформы в пространстве.

Итоговая программная система оказалась способной для управления движением роботизированной платформой на основе заданного пути.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Визильтер Ю. В. Обработка и анализ цифровых изображений с примерами на LabView и IMAQ Vision./ Ю. В Визильтер, С. Ю. Желтов, В. А. Князь// – М.: ДМК-Пресс, 2007. – 464 с.

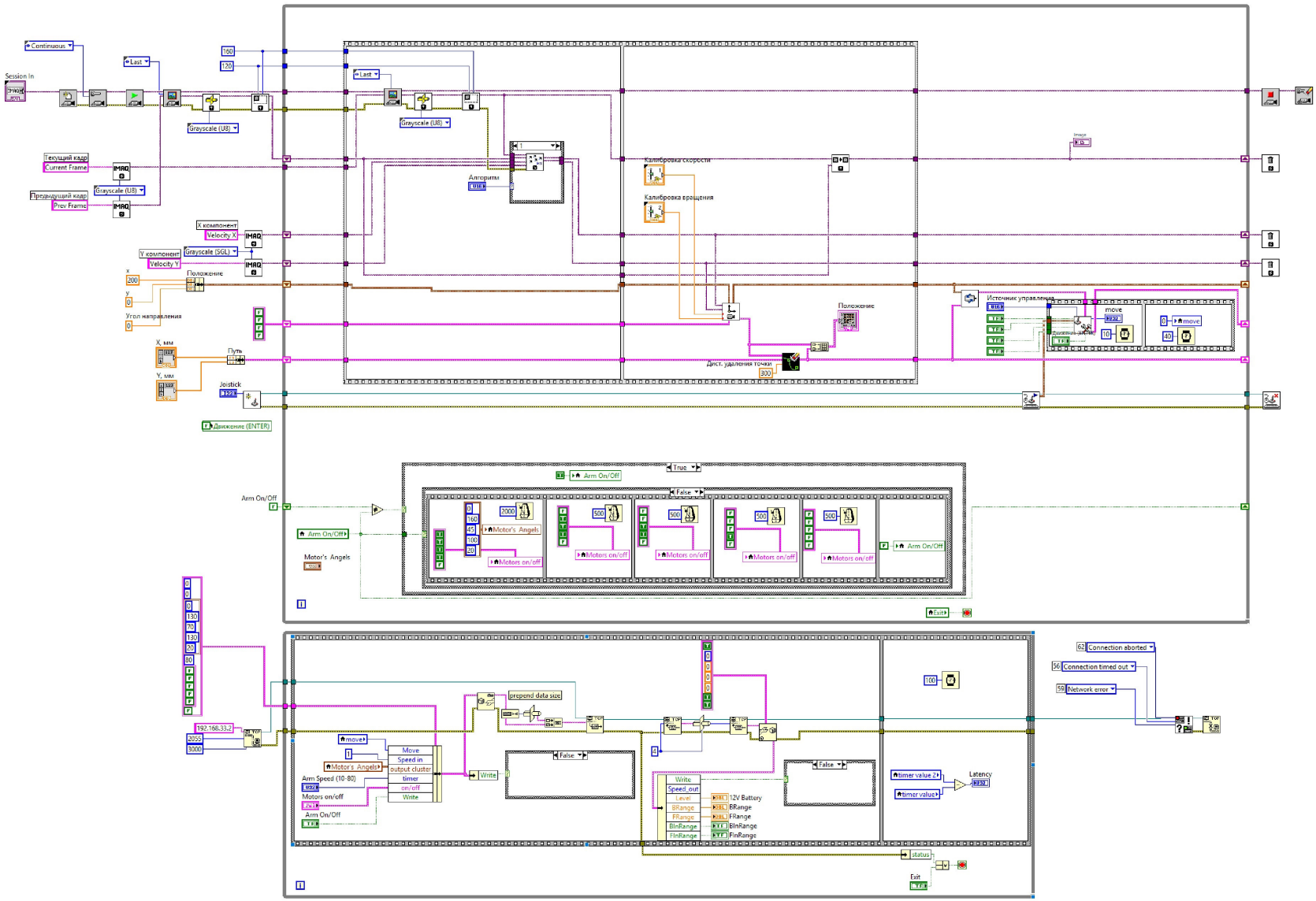
2. Белиновская Л. Г. основы машинного зрения в среде LabVIEW. Учебный курс./ Л. Г. Белиновская, Н. А. Белиновский// – М.: ДМК-Пресс, 2017. – 88 с.

3. Магда Ю. С. LabVIEW: Практический курс для инженеров и разработчиков./ Ю. С. Магда// – М.: ДМК-Пресс, 2011. – 208 с.

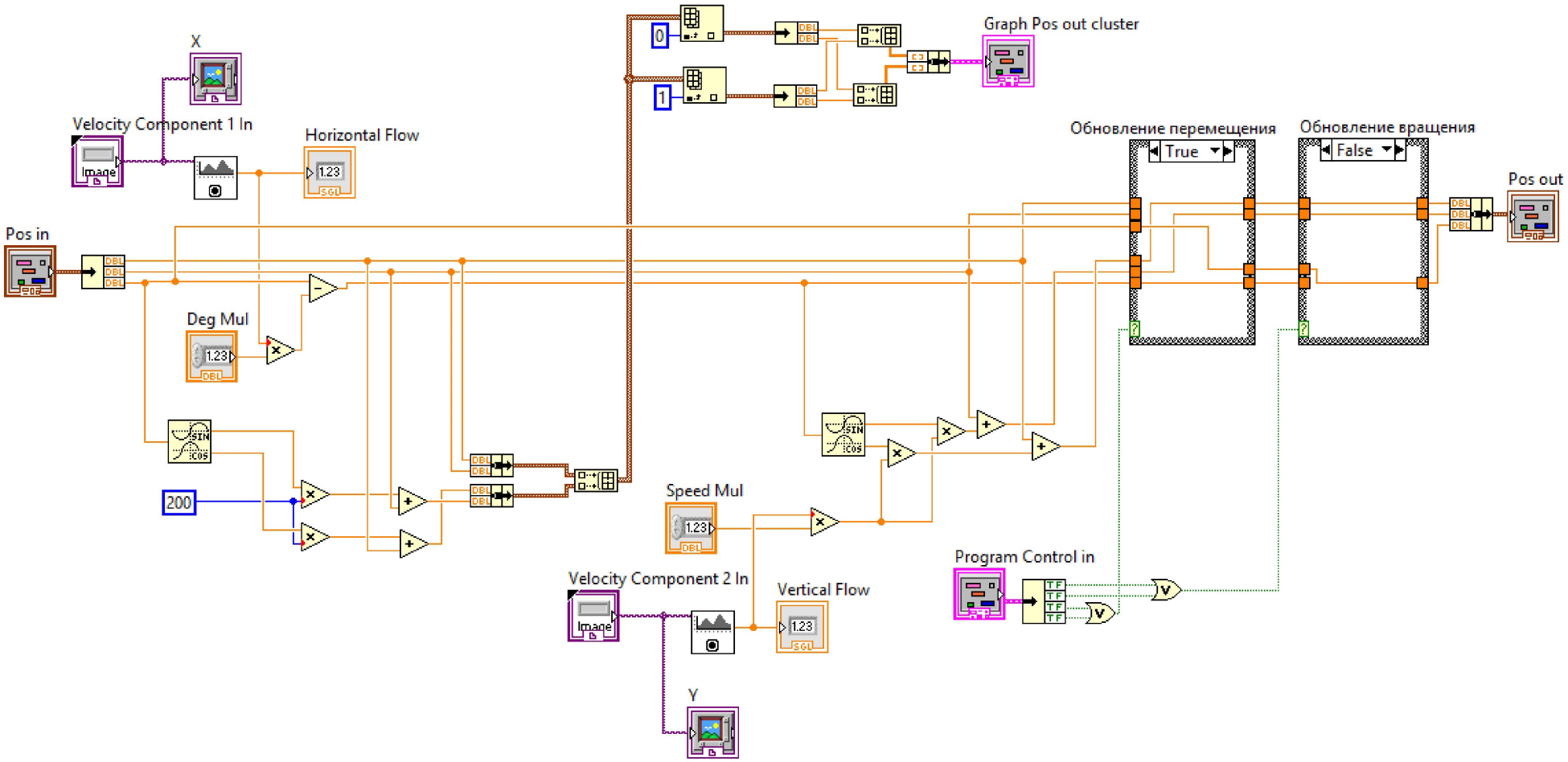
4. Кирсанов А. Ю. Дистанционный эксперимент на основе совмещения телекоммуникационных и измерительно-управляющих систем./ Ю. К. Евдокимов, А. Ю. Кирсанов, А. Ш. Салахова// – М.: Казанский государственный технический университет им. А.Н.Туполева, 2013. – 186 с.

5. Крапивин Д. М. Информационные системы в мехатронике и робототехнике./ Д. М. Крапивин// – М.: Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова, 2017. – 44 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Общий вид программного кода



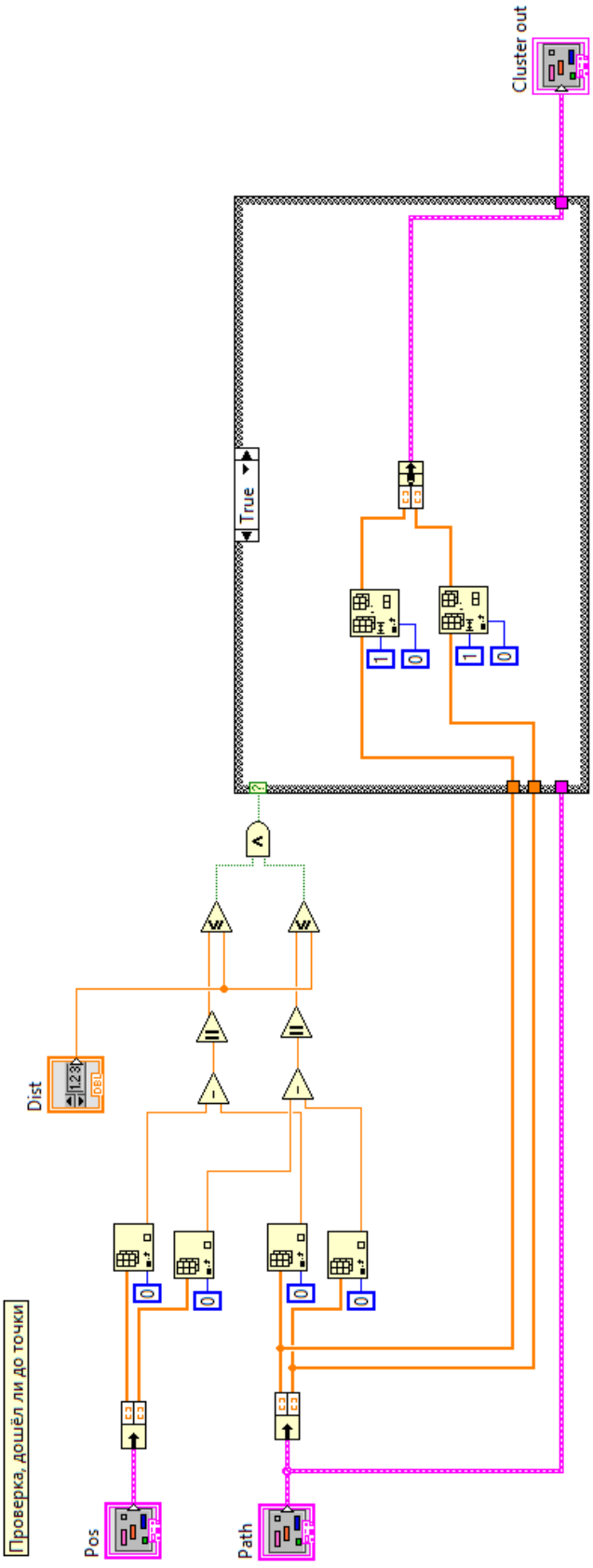
ПРИЛОЖЕНИЕ Б
(обязательное)
Код подпрограммы flow_speed.vi



ПРИЛОЖЕНИЕ В

(обязательное)

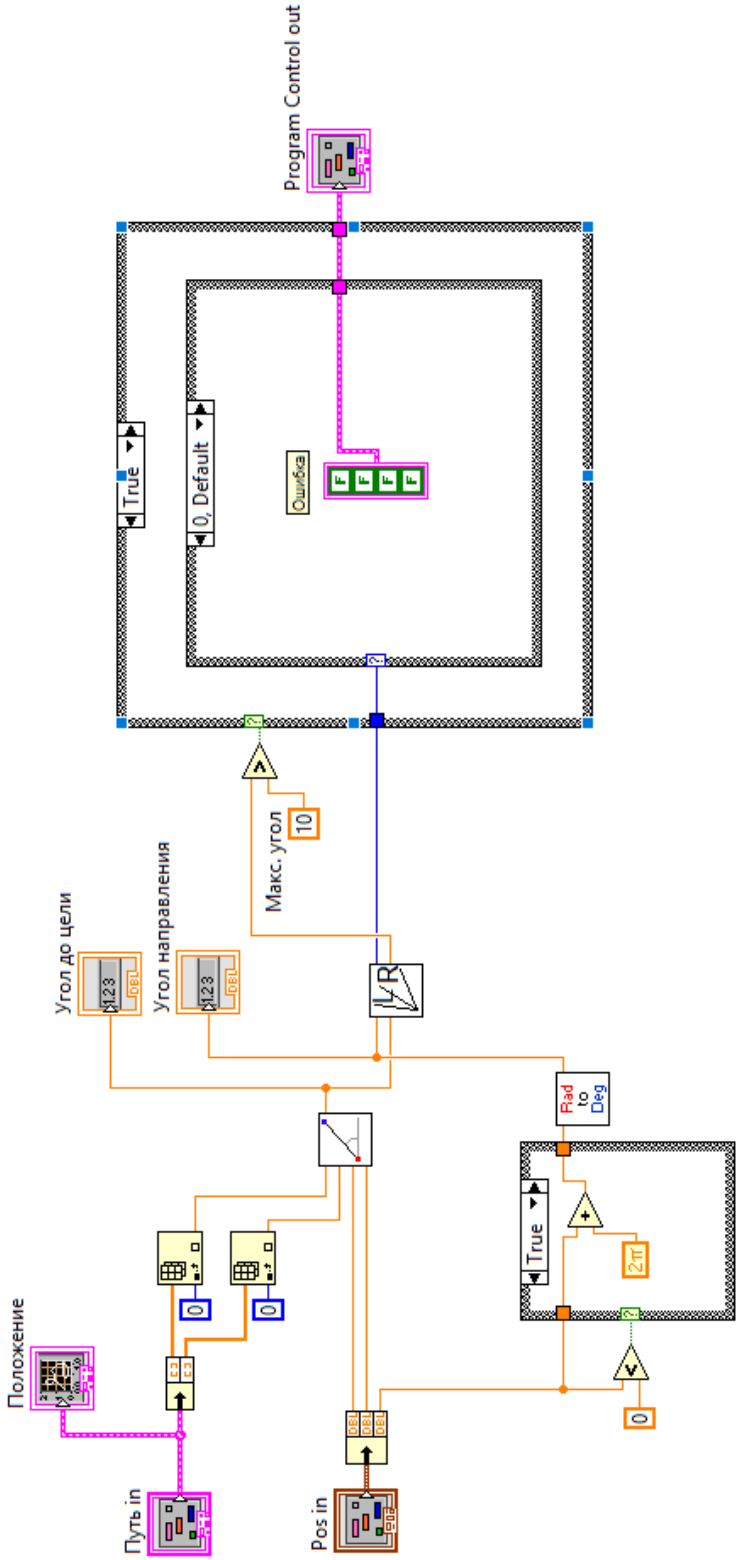
Код подпрограммы check_arrived_point.vi



ПРИЛОЖЕНИЕ Г

(обязательное)

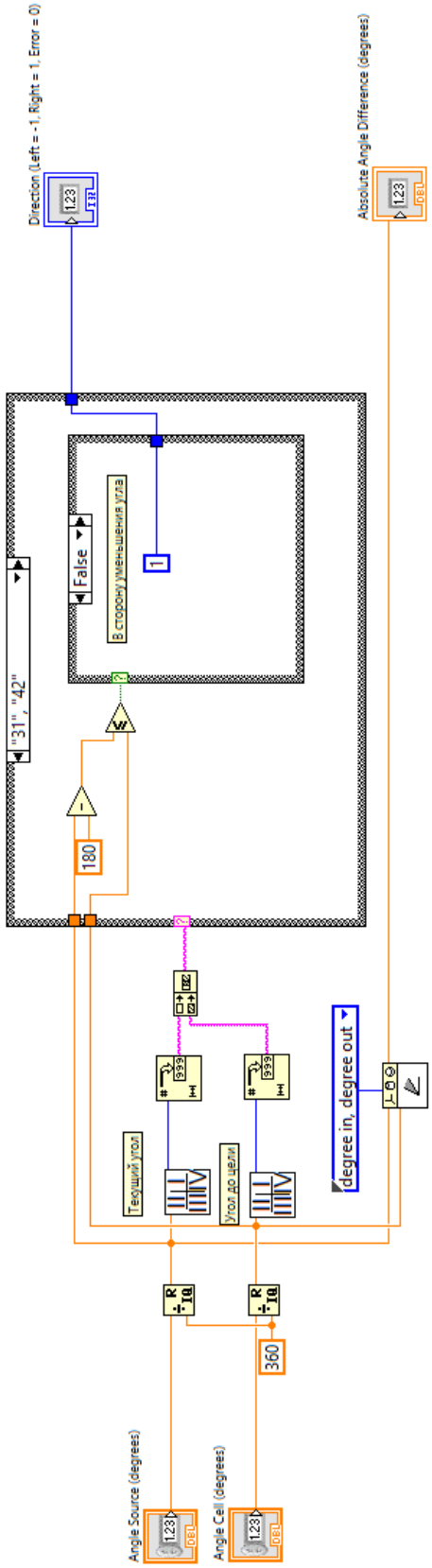
Код подпрограммы auto_moving.vi



ПРИЛОЖЕНИЕ Д

(обязательное)

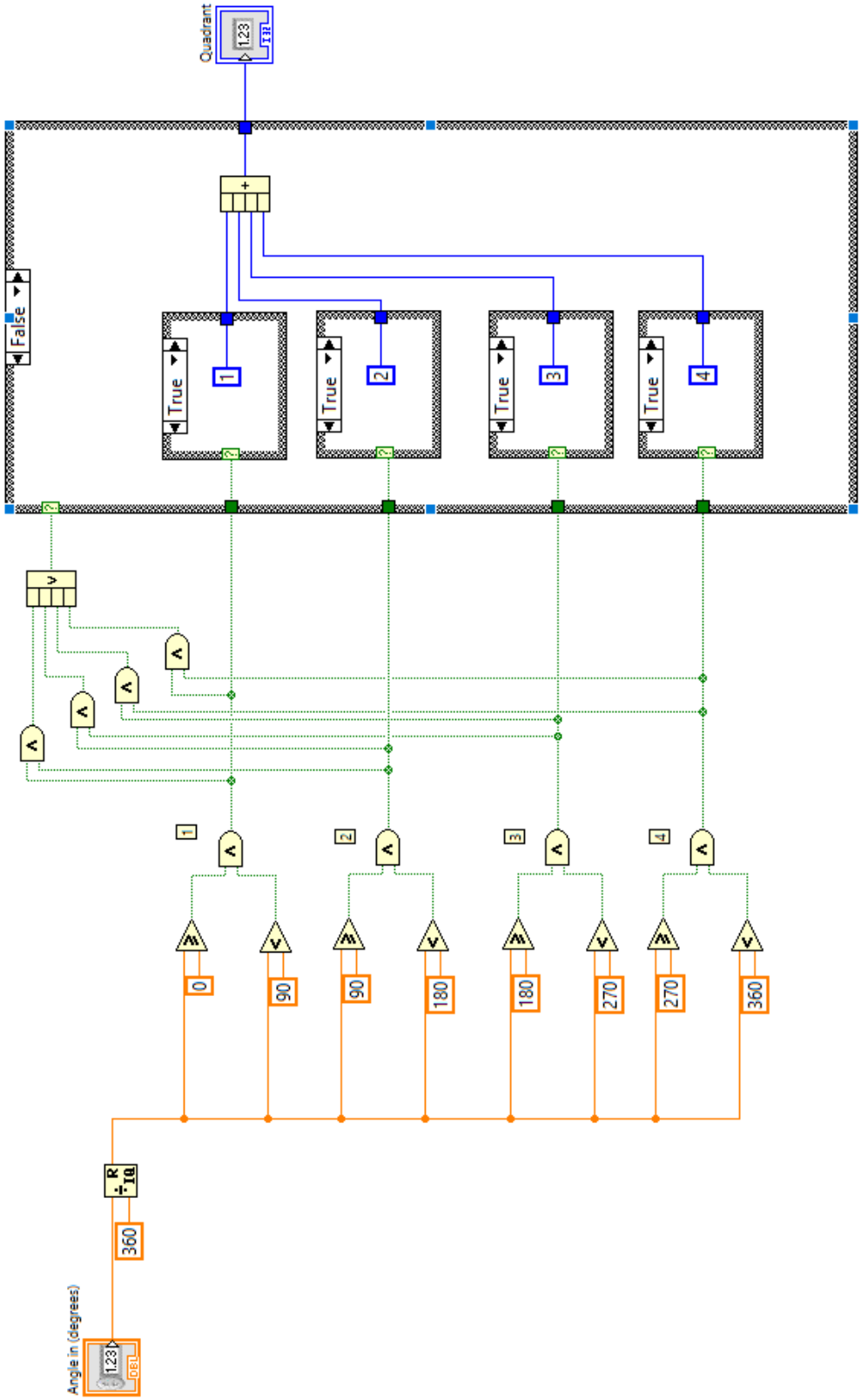
Код подпрограммы get_rotation_direction.vi



ПРИЛОЖЕНИЕ Е

(обязательное)

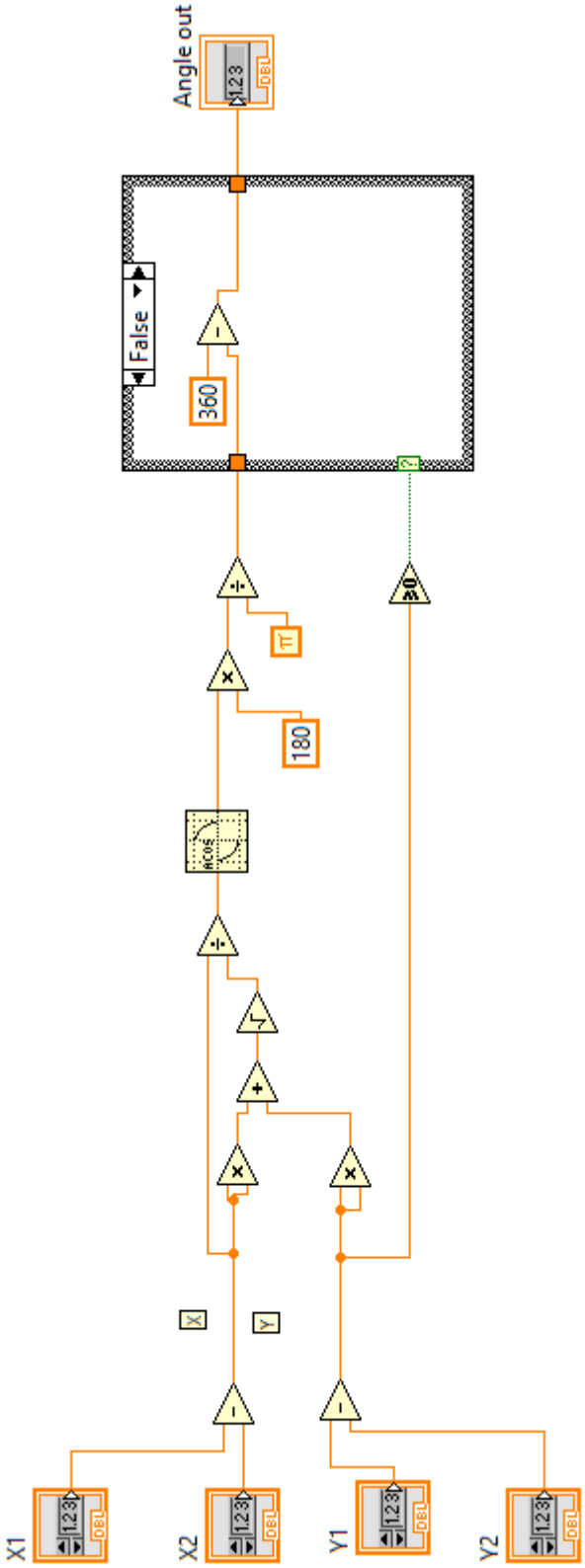
Код подпрограммы get_quadrant.vi



ПРИЛОЖЕНИЕ Ж

(обязательное)

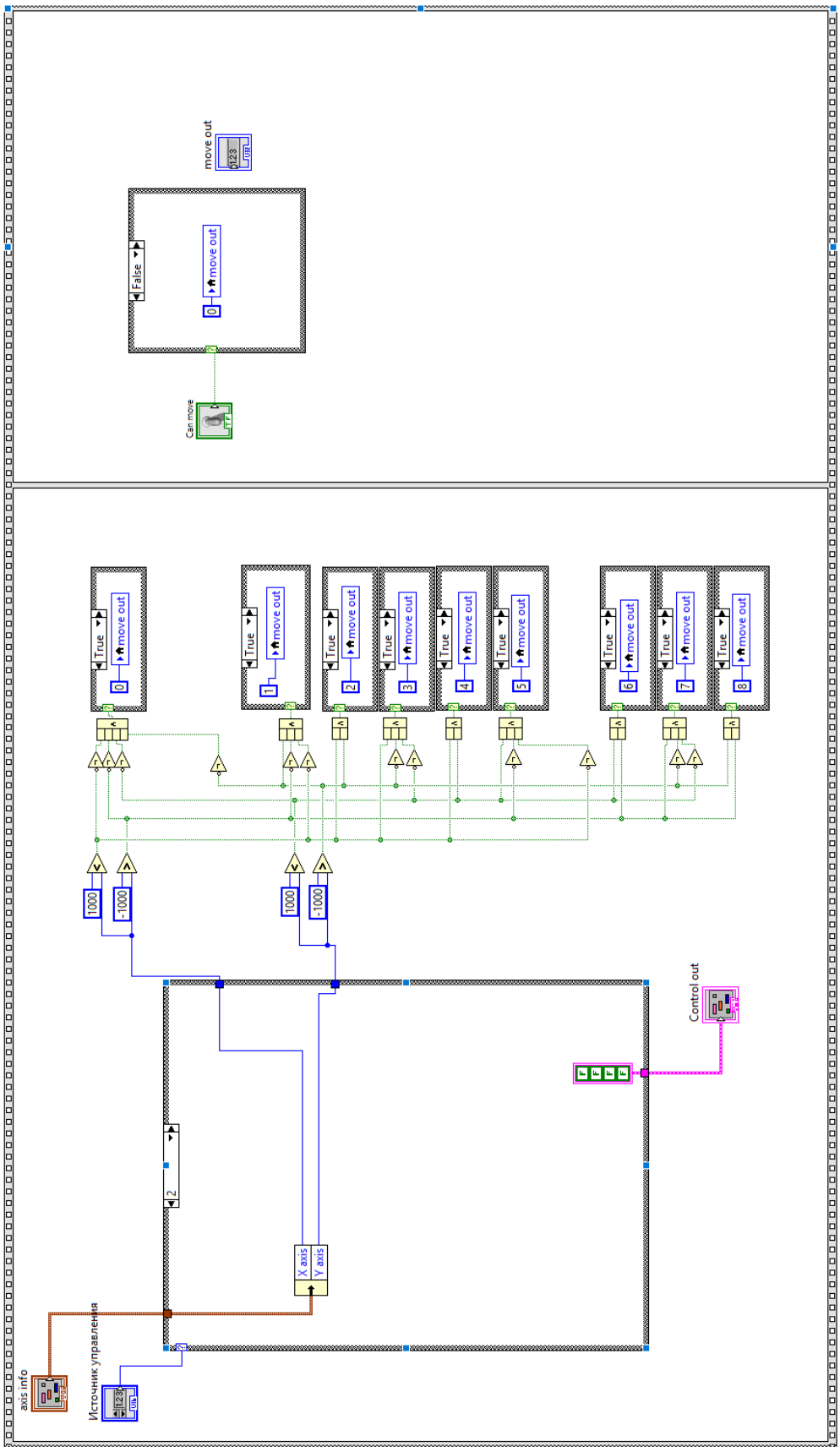
Код подпрограммы coord_to_angle.vi



ПРИЛОЖЕНИЕ З

(обязательное)

Код подпрограммы joy_move.vi



№ строки	Формат	Обозначение	Наименование	Кол.	Прим.
1					
2			Документация текстовая		
3					
4	А4	БР-02069964-11.03.04-08-18 ПЗ	Пояснительная записка	36	
5					
6			Документация графическая		
7					
8	А3	БР-02069964-11.03.04-08-18 ПЗ	Общий вид программного кода	1	Прил. А
9	А3	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы flow_speed.vi	1	Прил. Б
10	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы		
11			check_arrived_point.vi	1	Прил. В
12	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы auto_moving.vi	1	Прил. Г
13	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы		
14			get_rotation_direction.vi	1	Прил. Д
15	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы get_quadrant.vi	1	Прил. Е
16	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы coord_to_angle.vi	1	Прил. Ж
17	А4	БР-02069964-11.03.04-08-18 ПЗ	Код подпрограммы joy_move.vi	1	Прил. З
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

Изм.	Лист	№ докум.	Подпись	Дата	БР-02069964-11.03.04-08-18		
Разраб.		Кирсанов А. Е.		15.06.18	Разработка программного обеспечения роботизированной платформы предназначенной для обследования помещений	Лит.	Лист 1 / Листов 1
Пров.		Ильин М. В.		16.06.18			
Н. контр.		Шестёркина А. А.		20.06.18		МГУ им. Н. П. Огарёва ИЭС ЭНЭ 4.11 гр.	
Утв.		Беспалов Н. Н.		22.06.18			