Декомпозируем задачу финала на элементы, необходимые для ее решения:

1. Ровные проезды на заданное расстояние;
2. Повороты на заданный угол;
3. Движение по линии;
4. Определение перекрестков и поворотов при движении по линии;
5. Определение препятствий / границ полигона и расстояния до них.

Каждый элемент из этого списка удобнее реализовать с помощью подпрограмм (процедур и функций) либо в виде классов и объектов. Учитывая жесткое ограничение по времени на подготовку программы в 8 часов приоритетнее будет подготовить элементы в виде подпрограмм. Это не потребует проработки архитектуры классов и объектов и сэкономит время для реализации функционала.

Рассмотрим полный список процедур и функций, реализующих функционал описанных выше подзадач:

Далее будут описаны реализации этих процедур и функций в виде псевдокода.

MotorLeft(скорость) / MotorRight(скорость) – процедуры управления конкретными моторами. Левому и правому мотору сопоставляются конкретные порты контроллера, чтобы их не запоминать при дальнейшей работе. Кроме того, в этих процедурах можно реализовать ограничение максимальной скорости, чтобы не допускать пробуксовок колес по полю и отсекать заведомо большие скорости.

Входные параметры процедур – скорость вращения мотора в виде целого числа, способного принять как положительное, так и отрицательное значение. Положительная скорость заставляет мотор крутиться вперед, отрицательная – назад, нулевая – остановиться. Скорость может задаваться не в физических величинах (оборотах в минуту, радианах в секунду), а в процентах от максимальной мощности или в виде 8-битной скважности управляющего ШИМ-сигнала, в зависимости от аппаратной платформы.

Псевдокод одной из процедур:

```
максимальная_скорость = 100
…
ФУНКЦИЯ MotorLeft(скорость)
        знак = 1
        ЕСЛИ (скорость < 0 ) ТО  знак = -1 КОНЕЦ_ЕСЛИ
        ЕСЛИ ( абсолютное_значение(скорость) > макисимальная_скорость ) ТО
                скорость = максимальная_скорость * знак
        КОНЕЦ_ЕСЛИ
        Мотор_запустить( порт_левого_мотора, скорость )
КОНЕЦ_ФУНКЦИИ
```

MotorSincDrive(скорость_лев, скорость_прав) – процедура управления двумя моторами с синхронизацией по энкодерам, обеспечивающей равномерный их поворот и притормаживание одного мотора при нагрузке на другой. Эта процедура позволяет роботу двигаться прямолинейно, не отклоняясь. Для синхронизации используется алгоритм ПИД-регулятора: разница между показания энкодеров двух моторов принимаются за «ошибку регулирования». В этом случае регулятор стремится свести ошибку к нулю, то есть добиться отсутствия разницы между показания энкодеров, которая достигается при синхронном вращении моторов.

Входные параметры – желаемые скорости вращения левого и правого моторов в виде целых чисел, аналогичных скоростям из процедур **MotorLeft() / MotorRight()**.

Псевдокод процедуры:

```
время_прошлого_шага = 0
время_итерации = 0
КП_синх = 1
КИ_синх = 1
КД_синх = 1
ошибка_стар = 0
И_синх = 0
…
ФУНКЦИЯ MotorSincDrive(скорость_лев, скорость_прав)
        время_итерации = текущее_время – время_прошлого_шага
```

```
        время_прошлого_шага = текущее_время
        знак_лев = 1
        знак_прав = 1
        ЕСЛИ ( скорость_лев <>0 И скорость_прав <> 0) ТО
                ЕСЛИ ( скорость_лев < 0 ) ТО знак_лев = -1 КОНЕЦ_ЕСЛИ
                ЕСЛИ ( скорость_прав <0 ) ТО знак_прав = -1 КОНЕЦ_ЕСЛИ
                ошибка = энкодер_лев * скорость_прав / скорость_лев – энкодер_прав
                П_синх = КП_синх * ошибка
                И_синх = И_синх + КИ_синх*ошибка*время_итерации
                Д_синх = КД_синх*(ошибка – ошибка_стар) / время_итерации
                управляющее_воздействие = П_синх + И_синх + Д_синх
                коррект_скорость_лев = скорость_лев - управляющее_воздействие * знак_лев * знак_прав
                коррект_скорость_прав = скорость_прав + управляющее_воздействие
                MotorLeft(коррект_скорость_лев)
                MotorRight(коррект_скорость_прав)
        ИНАЧЕ
                MotorLeft(скорость_лев)
                MotorRight(скорость_прав)
        КОНЕЦ_ЕСЛИ
КОНЕЦ_ФУНКЦИИ
```

DistMotorSincDrive(скорость, расстояние) – процедура прямолинейного движения на заданное расстояние с заданной скоростью. Основана на предыдущей процедуре, но отслеживает пройденное расстояние и завершается при его прохождении. В приведенной ниже реализации процедуры отсутствуют плавный разгон и торможение. Если колеса робота при старте и остановке пробуксовывают, то можно ограничить максимальную скорость в процедурах **MotorLeft()** и **MotorRight()**.

Входные параметры – желаемая скорость движения робота и требуемое расстояние проезда. Скорость задается в виде целого числа аналогично входным параметрам функций MotorLeft() / MotorRight() / MotorSincDrive(). Расстояние задается в виде целого числа в миллиметрах.

Псевдокод процедуры:

```
диаметр_колеса = 82
передаточное_отношение_редуктора = 45
```

```
тики_энкодера = 12
…
ФУНКЦИЯ DistMotorSincDrive(скорость, расстояние)
        расстояние_в_тиках = абсолютное_значение(передаточное_отношение_редуктора * тики_энкодера * расстояние) / (число_Пи * диаметр_колеса)
        пройденное_расстояние_в_тиках = 0
        ЕСЛИ (скорость <> 0 ) ТО
                ПОКА (пройденное_расстояние_в_тиках < расстояние_в_тиках) ДЕЛАТЬ
                        пройденное_расстояние_в_тиках = абсолютное_значение(энкодер_лев + энкодер_прав) / 2
                        MotorSincDrive(скорость, скорость)
                КОНЕЦ_ПОКА
        КОНЕЦ_ЕСЛИ
        MotorSincDrive(0,0)
        И_синх = 0
        Сброс(энкодер_лев)
        Сброс(энкодер_прав)
КОНЕЦ_ФУНКЦИИ
```

MotorSincDriveToLine(скорость) – процедура проезда робота с синхронизацией моторов до черной линии. Позволяет роботу двигаться по белому полю до тех пор, пока оба датчика не увидят черную линию с дальнейшим проездом вперед, чтобы оказаться колесами на линии (доездом). Процедура может пригодиться при объезде препятствий.

Входные параметры – скорость вращения моторов во время движения. Задается в виде целого числа аналогично предыдущим процедурам.

Псевдокод процедуры:

```
расстояние_между_датчиками_и_колесами = 150
…
ФУНКЦИЯ MotorSincDriveToLine(скорость)
        ЕСЛИ (скорость <> 0 ) ТО
                ПОКА ( (яркость_лев + яркость_прав) > 2* граница_ЧБ
                        MotorSincDrive(скорость, скорость)
                КОНЕЦ_ПОКА
                DistMotorSincDrive(скорость, расстояние_между_датчиками_и_колесами)
```

```
        КОНЕЦ_ЕСЛИ
        MotorSincDrive(0, 0)
        И_синх = 0
        Сброс(энкодер_лев)
        Сброс(энкодер_прав)
КОНЕЦ_ФУНКЦИИ
```

AngleMotorSincRotate(скорость, угол) – процедура поворота на заданный угол. Основана на процедуре MotorSincDrive(), то есть во время поворота моторы вращаются синхронно: одно колесо вращается вперед, другое – назад с заданной скоростью. Поворот совершается вокруг центра оси установки колес. Такой поворот наиболее универсален и компактен.

Входные параметры – скорость вращения колес во время поворота и угол поворота. Скорость задается в виде целого числа аналогично предыдущим процедурам. Угол поворота задается в виде целого числа градусов. При этом положительное число означает поворот направо, отрицательное – налево.

Псевдокод процедуры:

```
диаметр_колеса = 82
передаточное_отношение_редуктора = 45
тики_энкодера = 12
ширина_колеи = 190
…
ФУНКЦИЯ AngleMotorSincRotate(скорость, угол)
        расстояние_в_тиках = (передаточное_отношение_редуктора * тики_энкодера * ширина_колеи * угол) / (диаметр_колеса * 360)
        пройденное_расстояние_в_тиках = 0
        знак = 1
        ЕСЛИ (угол < 0 ) ТО знак = -1 КОНЕЦ_ЕСЛИ
        ЕСЛИ (скорость <> 0 ) ТО
                ПОКА (пройденное_расстояние_в_тиках < расстояние_в_тиках) ДЕЛАТЬ
                        пройденное_расстояние_в_тиках = (абсолютное_значение(энкодер_лев) + абсолютное_значение(энкодер_прав) ) / 2
                        MotorSincDrive(знак * абсолютное_значение(скорость), -1 * знак * абсолютное_значение(скорость) )
                КОНЕЦ_ПОКА
        КОНЕЦ_ЕСЛИ
        MotorSincDrive(0,0)
```

```
        И_синх = 0
        Сброс(энкодер_лев)
        Сброс(энкодер_прав)
КОНЕЦ_ФУНКЦИИ
```

RotateToLine(скорость, направление, режим) – процедура поворота роботом в заданную сторону до тех пор, пока он не увидит датчиками освещенности черную линию. Используется для поворотов робота на перекрестках при движении по линии и позволяет совершить поворот и сразу оказаться датчиками на линии, готовым к дальнейшему движению.

Входные параметры – скорость вращения моторов при повороте робота, направление поворота и режим выравнивания датчиками на линии. Скорость задается в виде целого числа аналогично предыдущим процедурам. Направление задается в виде целых чисел 1 и -1, при этом число 1 означает поворот направо, число -1 – налево. Режим выравнивания задается числами -1, 0 и 1 и подразумевает выравнивание датчиками с разных сторон от линии (режим «0»), правым датчиком слева от линии (режим «1») или левым датчиком справа от линии (режим «-1»). Эти режимы выравнивания нужны для дальнейшего движения по линии одним или двумя датчиками. Движение по линии одним датчиком позволяет определять перекрестки с помощью другого.

Псевдокод процедуры:

```
граница_ЧБ = 50
...
ФУНКЦИЯ RotateToLine(скорость, направление, режим)
        AngleMotorSincRotate(скорость, направление * 40)
        MotorSincDrive(направление * абсолютное_значение(скорость), -1 * направление * абсолютное_значение(скорость) )
        ЕСЛИ (направление > 0) ТО
                ЕСЛИ (режим = 0) ТО
                        ПОКА (яркость_прав > 1.1 * граница_ЧБ) ДЕЛАТЬ
                        КОНЕЦ_ПОКА
                        ПОКА (яркость_прав < 0.9 * граница_ЧБ) ДЕЛАТЬ
                        КОНЕЦ_ПОКА
                КОНЕЦ_ЕСЛИ
                ЕСЛИ (режим = -1) ТО
                        ПОКА (яркость_лев > 1.1 * граница_ЧБ) ДЕЛАТЬ
                        КОНЕЦ_ПОКА
```

```
                ПОКА (яркость_лев < 0.9 * граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
            КОНЕЦ_ЕСЛИ
            ЕСЛИ (режим = 1) ТО
                ПОКА (яркость_прав > граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
            КОНЕЦ_ЕСЛИ
        КОНЕЦ_ЕСЛИ
        ЕСЛИ (направление < 0) ТО
            ЕСЛИ (режим = 0) ТО
                ПОКА (яркость_лев > 1.1 * граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
                ПОКА (яркость_лев < 0.9 * граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
            КОНЕЦ_ЕСЛИ
            ЕСЛИ (режим = -1) ТО
                ПОКА (яркость_лев > граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
            КОНЕЦ_ЕСЛИ
            ЕСЛИ (режим = 1) ТО
                ПОКА (яркость_прав > 1.1 * граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
                ПОКА (яркость_прав < 0.9 * граница_ЧБ) ДЕЛАТЬ
                КОНЕЦ_ПОКА
            КОНЕЦ_ЕСЛИ
        КОНЕЦ_ЕСЛИ
        MotorSincDrive(0,0)
КОНЕЦ_ФУНКЦИИ
```

LineFollow(скорость, режим) – процедура движения по линии на двух датчиках освещенности. Позволяет роботу двигаться вдоль черной линии на белом фоне (именно такие линии встречаются на полигоне финальной задачи). Для выравнивания на линии используется алгоритм ПИД-регулятора. В режиме «0» разница между показания двух датчиков принимается за «ошибку

регулирования». В этом случае регулятор стремится свести ошибку к нулю, то есть добиться одинаковых показаний на обоих датчиках, которые сравняются при выравнивании робота строго над линией.

В режимах «-1» и «1» в качестве ошибки регулирования принимается разница между текущими и желаемыми показаниями датчика. Желаемыми показаниями устанавливаются показания датчика на границе черной линии и белого фона (так называемый «серый»). Если датчик и робот отклоняются в сторону черной линии, то показания датчика освещенности снижаются, если в сторону белого фона – растут.

Входные параметры – скорость вращения моторов при движении робота строго по линии и режим выравнивания. Скорость задается в виде целого числа аналогично входным параметрам предыдущих процедур. Режим задается в виде числе -1, 0 или 1 аналогично процедуре **RotateToLine()**.

Псевдокод процедуры:

```
время_прошлого_шага = 0
время_итерации = 0
КП_лин = 1
КИ_лин = 1
КД_лин = 1
ошибка_стар = 0
И_лин = 0
…
ФУНКЦИЯ LineFollow(скорость, режим)
        время_итерации = текущее_время – время_прошлого_шага
        время_прошлого_шага = текущее_время
        ошибка = 0
        ЕСЛИ (режим = 0) ТО
                ошибка = яркость_лев – яркость_прав
        КОНЕЦ_ЕСЛИ
        ЕСЛИ (режим = 1) ТО
                ошибка = яркость_прав - граница_ЧБ
        КОНЕЦ_ЕСЛИ
        ЕСЛИ (режим = -1) ТО
                ошибка = граница_ЧБ - яркость_лев
        КОНЕЦ_ЕСЛИ
```

```
        П_лин = КП_лин * ошибка
        И_лин = И_лин + КИ_лин * ошибка * время_итерации
        Д_лин = КД_лин * (ошибка – ошибка_стар) / время_итерации
        управляющее_воздействие = П_лин + И_лин + Д_лин
        скорость_лев = скорость + управляющее_воздействие
        скорость_прав = скорость - управляющее_воздействие
        MotorLeft(скорость_лев)
        MotorRight(скорость_прав)
КОНЕЦ_ФУНКЦИИ
```

DistLineFollow(скорость, расстояние, режим) – процедура движения робота по линии на заданное расстояние. Позволяет проехать по линии заданное расстояние. Может быть полезна при объезде препятствий.

Входные параметры – скорость вращения моторов при движении по линии, расстояние проезда и режим выравнивания. Скорость и режим выравнивания задаются в виде целых числе аналогично предыдущим процедурам, расстояние задается в виде целого числа в миллиметрах.

Псевдокод процедуры:

```
ФУНКЦИЯ DistLineFollow(скорость, расстояние, режим)
        расстояние_в_тиках = абсолютное значение(передаточное_отношение_редуктора * тики_энкодера * расстояние) / (число_Пи * диаметр_колеса)
        пройденное_расстояние_в_тиках = 0
        ЕСЛИ (скорость <> 0 ) ТО
                ПОКА (пройденное_расстояние_в_тиках < расстояние_в_тиках) ДЕЛАТЬ
                        пройденное_расстояние_в_тиках = абсолютное_значение(энкодер_лев + энкодер_прав) / 2
                        LineFollow(скорость, режим)
                КОНЕЦ_ПОКА
        КОНЕЦ_ЕСЛИ
        LineFollow(0, режим)
        И_лин = 0
КОНЕЦ_ФУНКЦИИ
```

LineFollowToCrossroad(скорость, режим) – процедура движения робота по линии до перекрестка. Обеспечивает проезд до перекрестка в заданном режиме выравнивания и остановку датчиками на перекрестке.

В режиме «0» (выравнивание по линии обоими датчиками) перекресток определяется при снижении яркости на обеих датчиках.

В режиме «1» (выравнивание правым датчиком слева от линии) перекресток определяется при снижении яркости на левом датчике.

В режиме «-1» (выравнивание левым датчиком справа от линии) перекресток определяется при снижении яркости на правом датчике.

Входные параметры – скорость вращения моторов при движении по линии и режим выравнивания. Оба параметры задаются в виде целых чисел аналогично предыдущим процедурам.

Псевдокод процедуры:

```
ФУНКЦИЯ LineFollowToCrossroad(скорость, режим)
        ЕСЛИ (скорость <> 0 ) ТО
                флаг = 0
                ПОКА (ФЛАГ < 1) ДЕЛАТЬ
                        LineFollow(скорость, режим)
                        ЕСЛИ (режим = 0 И (яркость_лев + яркость_прав) < 2 * граница_ЧБ ) ТО
                                флаг = 1
                        КОНЕЦ_ЕСЛИ
                        ЕСЛИ (режим = 1 И яркость_лев < граница_ЧБ) ТО
                                флаг = 1
                        КОНЕЦ_ЕСЛИ
                        ЕСЛИ (режим = -1 И яркость_прав < граница_ЧБ) ТО
                                флаг = 1
                        КОНЕЦ_ЕСЛИ
                КОНЕЦ_ПОКА
        КОНЕЦ_ЕСЛИ
        LineFollow(0, режим)
        И_лин = 0
КОНЕЦ_ФУНКЦИИ
```

LineFollowToCrossroadAndRotatePrepare(скорость, режим) – процедура движения робота по линии до перекрестка с подготовкой к повороту на нем. Отличается от предыдущей процедуры тем, что останавливает робота на перекрестке не датчиками, а серединой оси установки колес.

Процедура состоит из трех логических действий: проезда до перекрестка, движения вперед с синхронизацией моторов на 30 мм для проезда перекрестка и проезда по линии на расстояние, необходимое, чтобы оказаться колесами на перекрестке (доезд).

Входные параметры – скорость вращения моторов при движении по линии и режим выравнивания. Оба параметры задаются в виде целых чисел аналогично предыдущим процедурам.

Псевдокод процедуры:

```
расстояние_между_датчиками_и_колесами = 150
...
ФУНКЦИЯ LineFollowToCrossroadAndRotatePrepare(скорость, режим)
        LineFollowToCrossroad(скорость, режим)
        DistMotorSincDrive(скорость, 30)
        DistLineFollow(скорость, расстояние_между_датчиками_и_колесами - 30, режим)
КОНЕЦ_ФУНКЦИИ
```

UltraSonicFilt(количество_итераций) – функция фильтрации показаний датчика расстояния. Датчик расстояния не имеет аппаратной фильтрации (в отличие от датчика освещенности) и может выдавать значительно зашумленные показания – переотраженные эхо-сигналы, резко меняющиеся значения и т.д. Для гарантированного определения препятствий необходимо избавиться от этих шумов (помех), для чего и предназначена эта функция. Она основана на алгоритме экспоненциального фильтра, как наиболее быстрореализуемого.

Функция делает несколько измерений датчиком и возвращает сглаженное показание, не допуская резких отклонений от предыдущего (усредненного) значения. Функция возвращает одно значение и может применяться после остановки движения робота. Это минимально необходимая реализация функции; для более эффективного решения финальной задачи может потребоваться модернизировать ее для того, чтобы фильтровать показания датчика без остановки робота.

Входные параметры – количество итераций считывания показаний датчика. Задается в виде целого положительного числа. Каждая итерация считывания занимает некоторое время, поэтому слишком большое количество итераций может остановить робота на значительное время.

Возвращаемое значение – отфильтрованное расстояние до объекта, которое видит датчик. Расстояние возращается в виде дробного числа с тем же значением, что и показания датчика, то есть, если датчик считывает расстояние в миллиметрах, то и функция возвращает миллиметры.

Псевдокод функции:

```
КФ = 0.98
...
ФУНКЦИЯ UltraSonicFilt(количество_итераций)
        счётчик = 1
        фильтрованные_показания = считать_датчик(порт_датчика_расстояния)
        ПОКА (счетчик < количество_итераций)
                фильтрованные_показания = фильтрованные_показания * КФ + (1 - КФ) * считать_датчик(порт_датчика_расстояния)
                счётчик = счётчик + 1
        КОНЕЦ_ПОКА
        ВЕРНУТЬ фильтрованные_показания
КОНЕЦ_ФУНКЦИИ
```

Использование нескольких режимов движения по линии и выравнивания на ней достаточно громоздко. Его можно избежать, разделив каждую из этих процедур на три отдельных.

Реализовав все эти подпрограммы, можно собрать итоговую программу, позволяющую решить финальную задачу. В дополнение к уже обозначенным зонам на полигоне выделим несколько ключевых точек, в которых может оказаться робот в процессе выполнения задания (рисунок 1).

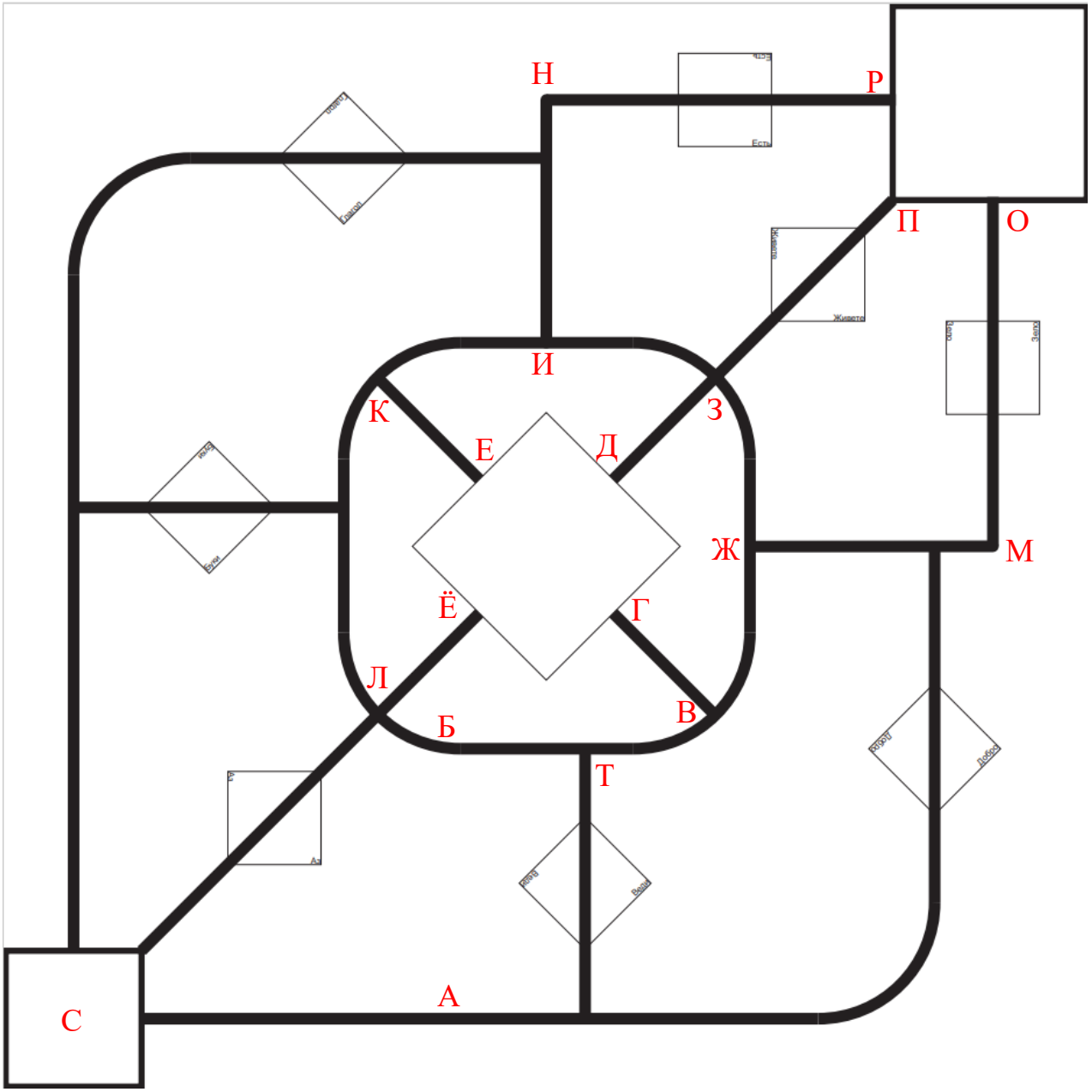

Рисунок 1. Ключевые точки на полигоне

Рассмотрим одну из возможных последовательностей действий робота, выполняющую финальную задачу:

№ п.п.	Описание	Псевдокод
1.	Предварительная настройка параметров робота и программы.	максимальная_скорость = 100 время_прошлого_шага = 0 время_итерации = 0 Есть = 1 Живете = 1 Зело = 1 КП_синх = 1 КИ_синх = 1 КД_синх = 1 ошибка_стар = 0 И_синх = 0 КП_лин = 1 КИ_лин = 1 КД_лин = 1 ошибка_стар = 0 И_лин = 0 КФ = 0.98 граница_ЧБ = 50 диаметр_колеса = 82 передаточное_отношение_редуктора = 45 тики_энкодера = 12 расстояние_между_датчиками_и_колесами = 150 ширина_колеи = 190
2.	Выезд из зоны старта (точка С) в сторону зон установки препятствий «Веди» и «Добро» до линии.	DistLineFollow(50, 300)
3.	Движение по линии на 620 мм, чтобы оказаться между зонами установки препятствий «Аз» и «Веди» (в точке А).	DistLineFollow(50, 620)
4.	Поворот на 90 градусов в сторону зоны погрузки (в сторону точки Б).	AngleMotorSincRotate(50, -90)
5.	Проезд до линии, окружающей зону погрузки (в точку Б).	MotorSyncDriveToLine(50)

6.	Поворот направо до линии, чтобы оказаться на линии готовым для дальнейшего движения к перекрестку В; выгоднее всего двигаться по линии правым датчиком, а левым искать перекрестки.	RotateToLine(50, 1)
7.	Движение по линии до перекрестка В с выравниванием серединой оси установки колес на перекрестке.	LineFollowToCrossroadAndRotatePrepare(50, 1)
8.	Поворот налево до линии, чтобы оказаться готовым к движению в точку Г.	RotateToLine(50, -1, 0)
9.	Измерение расстояния до препятствия (зоны погрузки).	дистанция = UltraSonicFilt(100)
10.	Установка кузова в положение «верх».	Мотор_запустить (порт_серво, верх_положение)
11.	Проезд по линии в точку Г. Этот проезд может быть равным измеренному на предыдущем шаге расстоянию или на 10-20 мм быть меньше его, чтобы не задеть и не сдвинуть зону погрузки. В нашем примере используется скорость / мощность в 50%, которой недостаточно для смещения зоны.	DistLineFollow(50, дистанция, 0)
12.	Отъезд назад на 100 мм для комфортного дальнейшего разворота. Без этого отъезда робот будет упираться в механизм погрузки и разворот может не быть выполненным до конца.	DistMotorSincDrive(-50, 100)
13.	Разворот до линии между В и Г и выравнивание на ней.	RotateToLine(50, 1, 0)
14.	Движение до перекрестка В с последующим выравниванием на ней серединой установки колес. В коде приведен пример с синхронизацией моторов вместо движения по линии после обнаружения перекрестка, так как линия отсутствует на этом участке. При корректной настройке датчиков освещенности и равных их показаниях на белом фоне движение робота будет достотачно ровным, чтобы использовать процедуру **LineFollowToCrossroadAndRotatePrepare()**. На данном этапе удобнее будет создать новую функцию для выравнивания на перекрестке и использовать ее в дальнейшем в подобных случаях.	ФУНКЦИЯ AltLineFollowToCrossroadAndRotatePrepare(скорость, режим) LineFollowToCrossroad(50, режим) DistMotorSincDrive(50, расстояние_между_датчиками_и_колесами) MotorSincDrive(0,0) КОНЕЦ_ФУНКЦИИ AltLineFollowToCrossroadAndRotatePrepare(50, 0)

15.	Поворот налево до линии, чтобы робот был направлен в сторону перекрестка Ж. Так как на этот перекресток не полный, а имеет только ответвление вправо, то удобнее выравниваться и двигаться левым датчиком справа от линии.	RotateToLine(50, -1, -1)
16.	Движение по линии до перекрестка Ж с выравниванием колесами на нем.	LineFollowToCrossroadAndRotatePrepare(50, -1)
17.	Поворот направо в сторону точки М, проезд до нее и поворот налево в сторону точки О.	RotateToLine(50, 1, 1) AltLineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 0)
18.	Находясь в точке М необходимо проверить показания датчика расстояния, чтобы определить, находится ли препятствие в зоне «Зело». Если его там нет, то обозначаем проезд свободным. В дальнейшем будем доставлять объекты через эту зону. Проезд в точку З с направлением в сторону точки П, дальнейшие движения будут строиться от нее.	дистанция = UltraSonicFilt(100) ЕСЛИ (дистанция > 500) ТО Зело = 0 КОНЕЦ_ЕСЛИ RotateToLine(50, -1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, 0)

19.	Если проезд через зону «Зело» оказался занят препятствием, то необходимо совершить движение по точкам З-И-Н и измерить расстояние там. Если проезд между точками Н-Р свободен (нет препятствия в зоне «Есть»), то будем доставлять объекты через него. Если и этот проезд окажется занятым, то будем доставлять объекты через зону «Живете».	ЕСЛИ (Зело > 0) ТО
		  RotateToLine(50, -1, -1)
		  LineFollowToCrossroadAndRotatePrepare(50, -1)
		  RotateToLine(50, 1, -1)
		  AltLineFollowToCrossroadAndRotatePrepare(50, -1)
		  RotateToLine(50, 1, 0)
		  дистанция = UltraSonicFilt(100)
		  ЕСЛИ (дистанция > 500) ТО
		  Есть = 0
		  ИНАЧЕ
		  Живете = 0
		  КОНЕЦ_ЕСЛИ
		  RotateToLine(50, 1, 0)
		  LineFollowToCrossroadAndRotatePrepare(50, 0)
		  RotateToLine(50, -1, 0)
		  LineFollowToCrossroadAndRotatePrepare(50, 0)
		  RotateToLine(50, -1, 0)
		КОНЕЦ_ЕСЛИ

20.	Выполняем проезд и выгрузку первого объекта через зону без препятствия, возвращаемся в точку З, поворачиваемся в сторону точки Д. На этом этапе удобнее создать отдельную процедуру для выгрузки объектов и возврата в точку З. Входным параметром процедуры будет номер сектора для выгрузки (в зависимости от свободных зон проездов нумерация будет отличаться).	ФУНКЦИЯ ReleaseObject(сектор) ЕСЛИ (Зело = 0) ТО RotateToLine(50, 1, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 1) AltLineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) ЕСЛИ (сектор < 4) ТО DistMotorSincDrive(50, 250) AngleMotorSincRotate(50, 90*(сектор-2)) Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) AngleMotorSincRotate(50, -90*(сектор-2)) DistMotorSincDrive(-50, 250) ИНАЧЕ Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) КОНЕЦ_ЕСЛИ RotateToLine(50, -1, 0) RotateToLine(50, -1, -1) AltLineFollowToCrossroadAndRotatePrepare(50, -1) RotateToLine(50, 1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, -1, 0) ИНАЧЕ ЕСЛИ (Есть = 0) ТО RotateToLine(50, -1, -1) LineFollowToCrossroadAndRotatePrepare(50, -1) RotateToLine(50, 1, -1) AltLineFollowToCrossroadAndRotatePrepare(50, -1) RotateToLine(50, 1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) ЕСЛИ (сектор < 4) ТО DistMotorSincDrive(50, 250) AngleMotorSincRotate(50, -90*(2-сектор))

		Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) AngleMotorSincRotate(50, 90*(2сектор)) DistMotorSincDrive(-50, 250) ИНАЧЕ Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) КОНЕЦ_ЕСЛИ RotateToLine(50, 1, 0) RotateToLine(50, 1, 1) AltLineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, -1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, 0) ИНАЧЕ AltLineFollowToCrossroadAndRotatePrepare (50, 0) ЕСЛИ (сектор < 3) ТО DistMotorSincDrive(50, 300) AngleMotorSincRotate(50, -45*3+90*сектор) Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) AngleMotorSincRotate(50, 45*3-90*сектор) DistMotorSincDrive(-50, 300) ИНАЧЕ AngleMotorSincRotate(50, -45*5+90*сектор) Мотор_запустить (порт_серво, ниж_положение) Ждать(0.4 секунды) AngleMotorSincRotate(50, 45*5-90*сектор) КОНЕЦ_ЕСЛИ RotateToLine(50, 1, 0) RotateToLine(50, 1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0) КОНЕЦ_ЕСЛИ ReleaseObject(1)

21.	Повтор шагов 9-14 и погрузка объект из точки Д. Возврат в точку З в направлении точки П.	дистанция = UltraSonicFilt(100) Мотор_запустить (порт_серво, верх_положение) DistLineFollow(50, дистанция, 0) DistMotorSincDrive(-50, 100) RotateToLine(50, 1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0)
22.	В зависимости от вычисленной ранее пустой зоны установки препятствий проезд и выгрузка второго объект. Возврат в точку З в направлении точки Д.	ReleaseObject(2)
23.	Проезд в точку К, погрузка объекта из точки Е, возврат в точку З.	RotateToLine(50, 1, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 0) дистанция = UltraSonicFilt(100) Мотор_запустить (порт_серво, верх_положение) DistLineFollow(50, дистанция, 0) DistMotorSincDrive(-50, 100) RotateToLine(50, 1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, -1) LineFollowToCrossroadAndRotatePrepare(50, -1) RotateToLine(50, 1, 0)
24.	Выполняем проезд и выгрузку третьего объекта через зону без препятствия, возврат в точку З, поворот в сторону точки Д.	ReleaseObject(3)
25.	Проезд в точку Л, погрузка объекта из точки Е, возврат в точку З.	RotateToLine(50, 1, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) RotateToLine(50, -1, 0) дистанция = UltraSonicFilt(100) Мотор_запустить (порт_серво, верх_положение) DistLineFollow(50, дистанция, 0) DistMotorSincDrive(-50, 100) RotateToLine(50, 1, 0) AltLineFollowToCrossroadAndRotatePrepare(50, 0) RotateToLine(50, 1, -1) LineFollowToCrossroadAndRotatePrepare(50, -1) LineFollowToCrossroadAndRotatePrepare(50, -1)

		RotateToLine(50, 1, 0)
26.	Выполняем проезд и выгрузку четвертого объекта через зону без препятствия, возврат в точку З, поворот в сторону точки Д.	ReleaseObject(4)
27.	Остался только возврат в зону старта / финиша с точки З. Для этого можно проехать перекрестки Ж и Т, продолжить движение по линии на 220 миллиметров до точки Б, повернуть к точке А, выровняться в ней по линии и доехать в зону финиша.	RotateToLine(50, -1, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) LineFollowToCrossroadAndRotatePrepare(50, 1) DistLineFollow(50, 220, 1) AngleMotorSincRotate(50, -90) MotorSincDriveToLine(50) DistMotorSincDrive(50, расстояние_между_датчиками_и_колесами) RotateToLine(50, 1, 0) LineFollowToCrossroadAndRotatePrepare(50, 0) DistMotorSincDrive(50, 200)

Программа требует отладки для эффективной работы. Все переменные, влияющие на поведение робота и требующие подбора, указаны в таблице выше в пункте 1. Рассмотрим их и процесс их подбора:

Параметр	Описание и метод подбора
максимальная_скорость = 100	Максимально допустимая скорость / мощность вращения моторов. Не может иметь отрицательного значения. Если колеса робота проскальзывают при начале и окончании движений и поворотов, то ее можно уменьшить.
время_прошлого_шага = 0 время_итерации = 0 Зело = 1 Есть = 1 Живете = 1 ошибка_стар = 0 И_синх = 0 И_лин = 0	Эти глобальные переменные должны быть созданы в самом начале программы и будут хранить значения в течение всего ее выполнения. Переменные «Зело», «Есть» и «Живете» изначально хранят ненулевые положительные значения, означающие наличие препятствий в соответствующих зонах. По мере выполнения задания робот проверит зоны и обнулит некоторые из переменных, обозначая свободные проезды.
КП_синх = 1 КИ_синх = 1 КД_синх = 1	Коэффициенты ПИД-регулятора для синхронизации моторов. Не могут иметь отрицательные значения. Могут иметь дробные значения. Изначально можно оставить переменные КИ_синх и КД_синх нулевыми, отключив интегральную и дифференциальную части регулятора. Если при этом робот значительно дергается при движении с синхронизацией моторов, то можно снизить КП_синх, если плавно

	«рыскает» и отклоняется далеко от изначального направления, то КП_синх можно повысить. Чтобы снизить «рыскание» можно увеличивать КД_синх, чтобы снизить отклонение от изначального направления – повысить КИ_синх. Для движения на низкой скорости может хватить только пропорциональной части, то есть КИ_синх и КД_синх могут остаться нулевыми.
КП_лин = 1 КИ_лин = 1 КД_лин = 1	Коэффициенты ПИД-регулятора для движения по линии. Не могут иметь отрицательные значения. Могут иметь дробные значения. Изначально можно оставить переменные КИ_лин и КД_лин нулевыми, отключив интегральную и дифференциальную части регулятора. Если при этом робот значительно дергается при движении по линии, то можно снизить КП_лин, если робот не успевает повернуть на резких поворотах линии, то КП_лин можно повысить. Чтобы снизить «рыскание» робота вдоль линии можно увеличивать КД_лин, чтобы ускорить плавное выравнивание на линии – повысить КИ_лин. Для движения на низкой скорости может хватить только пропорциональной части, то есть КИ_лин и КД_лин могут остаться нулевыми.
КФ = 0.98	Коэффициент фильтрации показаний датчика расстояния. Может иметь значение от 0 до 1. Чем выше значение коэффициента, тем сильнее будут сглаживаться «всплески» показаний, но тем медленнее будет обновляться отфильтрованное значение (что может потребовать большего количества итераций для адекватной фильтрации).
граница_ЧБ = 50	Показания датчиков освещенности на границе черной линии и белого фона. Вычисляется как среднее арифметическое между показанием на белом и показанием на черном фоне. Если показания левого и правого датчиков значительно отличаются, то может потребоваться создать две разных переменных, хранящих пограничные значения каждого датчика. В этом случае потребуется изменить исходный код функций, обращающихся к этой переменной.
передаточное_отношение_редуктора = 45 тики_энкодера = 12 расстояние_между_датчиками_и_колесами = 150 ширина_колеи = 190 диаметр_колеса = 82	Геометрические и механические параметры робота и его частей: колес, редукторов, энкодеров и т.д. Измеряются непосредственно на роботе или уточняются в документации / у организаторов.