Для реализации алгоритма обхода препятствий роботом в симуляторе turtlesim через ROS2 я решил сначала визуализировать препятствия для более удобной отладки моего решения поставленной задачи. Ввиду того, что я уже был знаком с некоторыми концепциями ROS2 на начало его изучения в рамках дисциплины, я принял решение о реализации движения черепахи путём создания пакета ROS2 с некоторыми нодами внутри, решение написано на языке Python.

Внутри пакета есть 4 ноды, из которых:

* 1 является тестированием предоставленного решения перемещения черепахи (***test\_zaicev***)
* 1 является дополнением этого решения с поворотом на целевой угол (***obstacles***)
* 1 является решением обхода перемещения на заданные координаты и целевой угол и обход препятствий через приближение к ним на определённое расстояние (***obstacles\_deepseek***)
* 1 является решением перемещения на заданные координаты и целевой угол и обход препятствий через приближение к первому препятствию с использованием алгоритма А\* для планирования пути (***obstacles\_final***)

В ходе работы для более удобного взаимодействия с пакетом и передачей целевых координат было принято решение создать топик, который будет принимать в себя несколько переменных. Существующие типы сообщений в ROS2 мне показались слишком сложными для отладки, необходимо набирать множество символом и сохранять жёсткую структуру сообщения для передачи информации в топик (Например, тип сообщения ***Float32*** или ***Float64***). Да, они конечно же универсальны для множества задач и проектов, но так как я решил сделать ноду, которая ожидает информацию из топика, то принял решение написать свой тип сообщения, который требовал бы меньше настроек сообщения для передачи сообщения путём ввода команды “***ros2 topic pub …***”. Мой тип сообщения находится в директории ***custom\_msgs*** и выполнен через CMake утилиту.

После того, как я закончил работу над свои решением, я понял, что мне нужно научиться правильно использовать систему контроля версий, поэтому я решил залить своё решение на GitHub через консоль т.к. по-другому это было делать гораздо сложнее.

Относясь непосредственно к своему алгоритму я могу выделить следующее:

* Алгоритм обхода препятствий имеет один путь прохода. Черепаха спокойно обойдет препятствия примерно такой формы, но не сможет зайти на все точки. Для более качественной реализации возможно имеет смысл разбить карту на графы, по которым черепаха должна перемещаться, чтобы выбрать ту точку препятствия, через которую она будет обходить их. и не задевать линии препятствий
* Если точка для перемещения черепахи находится не сзади препятствий, а спереди, она может запутаться и задеть линию препятствий. Это недопустимо, поэтому такой исход стоит учесть более детально в следующих своих работах
* Когда препятствие будет иметь более сильный разброс в своих точках, то возможно черепаха будет задевать их при переезде. Скорее всего это можно решить путём ввода пространства точек на линии от одной точки препятствия до другой, чтобы создать некоторую “запретную зону координат”, в таком случае черепаха всегда сможет достичь цели, если только она не находится слишком близко к линии препятствий
* Сейчас алгоритм учитывает только препятствия из 3-ёх точек. На деле же мы можем столкнуться с объектами более сложной формы и нам важно научиться обходить их. Сейчас такой возможности не представляется и черепаха может запутаться в своих перемещениях из-за движения к первой точке

Подводя итог всего сказанного выше, я научился:

1. Создавать свои топики
2. Создавать свой тип сообщения и использовать его в нодах
3. Разобрал детально структуру ноды и понимаю как используются отдельные функции внутри огромного класса контроллера
4. Кое-как обходить препятствия по их координатам
5. Контролировать версии проекта с использованием Git

В процессе своей работы я решил взять много помощи от нейросетей для более быстрой разработки своего решения, что несказанно мне помогло т.к. я мог понять тот ответ, что они мне дали для работы и исправлял его под свои нужды внутри редактора кода. Но в процессе этой работы я писал скорее всего слишком много запросов в неё и большую часть надеялся исправить уже готовый код, который будет сделан по моему запросу на конкретное улучшение или дополнение и скорее всего это и стало моей ошибкой при работе над решением. Я слишком решил довериться переписям от нейросети, когда возможно стоило сделать некоторые решения, которые я хотел добавить, самостоятельно, от этого я не смог сделать полноценно то, чего я хотел. В будущем это станет для меня уроком, не надеяться всецело на написание кода от ИИ, потому что иной раз они не в силах сделать то, что я пытаюсь объяснить и исправить самим свою же часть программы. Возможно, мне стоит больше углубиться в детализацию и планирование решения той задачи, которую ставлю перед собой, чтобы можно было эффективно сработаться с ней