ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

Регистрационный номер 12577

ИУ6

«Компьютерные системы» в составе секции «Информационные и компьютерные системы»

Разработка автопилота для подводного комплекса “Poseidon”

Авторы:

Исмагилов Герард Олегович

МБОУ ФМЛ № 31 города Челябинск, 10 класс

Научный руководитель:

Ловчиков Дмитрий Владимирович, заведующий лаборатории «Современные технологии»

МБОУ ФМЛ № 31 города Челябинск

Челябинск - 2022

Аннотация

Мониторинг и контроль подводно-устьевых систем добычи нефти производится с помощью датчиков и специального корабля, на борту которого находится подводный аппарат, спускаемый в воду для осмотра скважины. Мониторинг становится невозможен, если поверхность воды покрыта льдом или погодные условия не позволяют работать технике в открытом море и перемещаться между скважинами. Данную проблему решит создание автономного комплекса, который будет находиться под водой и осуществлять контроль скважин постоянно, вне зависимости от погодных условий. Ранее мы разработали АНПА, который может самостоятельно передвигаться под водой, но для функционирования комплекса требуется разработать автопилот для АНПА. Учитывая вышеизложенную актуальность, мы поставили перед собой цель: создать автопилот для малогабаритного подводного автономного аппарата для мониторинга подводных систем. Мы создали программный продукт для беспилотного движения подводного аппарата и выполнения миссий по сопровождению объекта и следованию вдоль нефтепровода. Чтобы обеспечивать движение и локализацию себя в пространстве подводный аппарат использует ряд датчиков: широкоугольную камеру высокого разрешения, датчик давления и цифровой компас. Этапы создания включают: 1. Определение требуемых функций для автопилота, 2. Написание функции автоматического удерживания глубины, 3. Написание функции автоматического удерживания курса, 4. Разработка алгоритма для передвижения вдоль нефтепровода, 5. Написание алгоритма распознавания направления стрелки, 6. Распознавание опасных объектов вдоль нефтепровода, 7. Разработка навигации с помощью Aruco маркеров, 8. Тестирование и исследование поведения робота в бассейне, 9. Проведение тестирования имитированного нефтепровода. Коды написанных функций, фото подтверждения результата находятся в тексте полной работы.

Содержание

ВВЕДЕНИЕ

Арктика — это кладезь минеральных ресурсов, огромные запасы нефтеуглеводородов, биологических ресурсов, превышающие таковые на суше в несколько раз. На арктической территории Якутии открыто порядка 400 месторождений полезных ископаемых. Из них почти 300 месторождений находятся в нераспределенном фонде недр, и главная причина их не освоения — это отсутствие достаточной инфраструктуры и техники. Для начала разработки этих месторождений нужны технические средства, которые могут заменить человека. В Арктике очень холодный климат и на воде присутствует постоянное движение льда, что не позволяет добывать нефть и газ традиционными методами. Перспективным методом является добыча с помощью подводно-устьевой системы. Технологический процесс выглядит следующим образом: на нефтедобывающей платформе или берегу находится центр управления добычей, а сами скважины находятся на дне моря и соединены с берегом за счет длинного подводного трубопровода и шланг кабеля, по которому передается информация об управлении оборудованием. По подводному трубопроводу поступает смесь газа, конденсата и воды с месторождения к центру управления. Скважины находятся на большом расстоянии друг от друга и требует постоянного визуального контроля над ней. В территориях, где поверхность воды не покрыта льдом, контроль осуществляется с помощью специального корабля, на котором установлен подводный аппарат (ПА). Корабль подплывает к скважине, опускает ПА, который производит осмотр, поднимает ПА и плывет к следующей скважине. Но если поверхность воды покрыта льдом, такой принцип визуального контроля становиться очень труд затратным и часто невозможным. Контроль могут осуществлять комплекс из автономных подводных аппаратов (АНПА). Аппараты будут находится под водой и постоянно осуществлять контроль над системой, передвигаться вдоль нефтепровода, осматривать скважину со всех сторон, записывать полученные данные в память и после передавать их диспетчеру. Линия передвижения представлена в Приложении И. Также АНПА

должны иметь базу для подзарядки и связи с диспетчером. Для выполнения функции мониторинга и контроля аппарат должен обладать следующими характеристиками: 1. Небольшие размеры 2. Маневренность 3. Конструкция аппарата, позволяющая производить быструю замену деталей. 4. Возможность двигаться в воде со скоростями, обеспечивающими движение против течения со скоростью не менее 1 морского узла. 5. Программируемый автопилот для управления моторами в автономном режиме. В 2020 году мы создавали собственный подводный аппарат Посейдон, для изготовления комплекса для подводного мониторинга, нам требуется разработать программируемый автопилот. Учитывая вышеизложенную актуальность, мы поставили перед собой **цель: создать автопилот для малогабаритного подводного автономного аппарата для мониторинга подводных систем.** Для достижения цели были поставлены следующий задачи:

- Определение требуемых функций для автопилота
- Написание функции автоматического удерживания глубины
- Написание функции автоматического удерживания курса
- Разработка алгоритма для передвижения вдоль нефтепровода
- Написание алгоритма распознавания направления стрелки
- Распознавание опасных объектов вдоль нефтепровода
- Разработка навигации с помощью Aruco маркеров
- Тестирование и исследование поведения робота в бассейне
- Проведение тестирования имитированного нефтепровода

Коды написанных функций, фото подтверждения результата находятся в тексте полной работы

ОСНОВНАЯ ЧАСТЬ

Создание АНПА Посейдон

Таблица 1 - Технические характеристики подводного аппарата

Общая масса, г	6000
Габариты, Д *Ш * В, см	40 * 35 * 25
Максимальная глубина действия, м	60 метров
Максимальное время работы, мин	50
Скорость передвижения в стоячей воде	не менее 2 морских узлов

Разработка конструкции корпуса

Свою работу мы начали с создания 3D модели аппарата, которая представлена в Приложении А. Мы работали в программе для 3-х мерного моделирования “КОМПАС 19”. Мы поэтапно создавали детали, объединяя их в подсборки и сборку. Модель включает в себя 30 деталей.

Корпус робота собирается из конструктора, который разрабатывался и производился нами. Наш конструктор – это набор деталей из поликарбоната разных размеров (Приложение Б), но с одинаковыми, универсальными отверстиями, что позволяет быстро изменять строение корпуса, а также дает возможность добавить множество дополнительных компонентов, не разрушая его первоначальной конструкции. Конструктор отличает наш аппарат от аналогов на рынке ПА, ведь при поломке детали корпуса, нам требуется изменить только одну часть конструктора, а не производить или закупать все его детали. Рама АНПА собирается из деталей конструктора, скрепленных между собой с помощью болтов на 3 мм и 5 мм.

Разработка герметичных движителей

Вода очень агрессивная среда, самая большая проблема — это большое давление под водой. Мы решили использовать технологию магнитной муфты, так как изготовленные движители получаются более надежными, поддаются

ремонту и могут работать в агрессивной среде. Собранный движитель состоит из мотора, постоянного тока, с надетой на него муфтой, напечатанной на 3д принтере. Она имеет 4 отверстия под магниты (20мм * 5мм * 5 мм). Мотор с муфтой помещается в защитную колбу с плотно закрывающейся крышкой. В первых версиях моторов, в качестве колбы, мы использовали литьевые трубы из орг. стекла, в трубу вставлялись и плотно приклеивались заглушки, но данный способ оказался не подходящий для использования под водой. Так как клей и заглушки пропускали воду, и мотор переставал работать. Поэтому, мы стали использовать преформы, ведь они имеют только одно отверстие, которое можно закрыть плотно закручивающейся крышкой. Конструкция движителя представлена в Приложении В.

После проведения испытаний, выяснили, что наши моторы не удовлетворяют требованиям, так как мощности мотора при напряжении 4V не хватало, а при увеличении напряжения магнитное кольцо с винтом слетало с защитного цилиндра, и винт не мог вращаться. Мы решили использовать AUV ROV 560KV 3KG подводный двигатель Thruster 300M. Цена такого движителя составляет 3500 рублей, а скорость 6000 оборотов в минуту (6000 RPM) и рабочая глубина до 300 метров под водой, что значительно лучше наших собственно произведенных движителей с магнитной муфтой, также Truster 300M имеет внешний заранее заизолированный драйвер, что позволяет уменьшить размеры и тепловыделение блока управления. Конструкция движителя представлена в Приложении Г.

Изготовление корпуса и блока управления

Электроника, управляющая движителями робота, помещена в специальную герметичную колбу для магистрального фильтра на 10 дюймов. Она имеет большое отверстие под хранение электроники и способна выдерживать большое давление и не пропускать воду. Для передвижения и анализа действий АНПА использует плату Asus Tinker Board, так как данная плата способна обрабатывать

видео в 1080p 30FPS, а также 2 gb RAM. На выбранной плате возможно реализовывать компьютерное зрение. Для управления моторам установлен микроконтроллер Arduino NANO. Для связи между контроллером и одноплатным компьютером, используется компактный USB 2.0 шнур. Конструкция блока электроники представлена в Приложении Д.

Отличие от аналогов

Наш аппарат имеет некоторые отличия от аналогов на рынке. 1) АНПА, которые применяются для решения реальных задач имеют форму торпеды, что увеличивает их радиус поворота, так как наш аппарат предназначен для автономного осмотра скважин, то его радиус поворота должен быть минимальным, для этого мы создали раму кубической формы. 2) Наш конструктор позволяет создавать корпус различных форм за минимальное время, каждая миссия уникальна из-за ее акватории или нужного оборудования. Конструктор помогает нам быстро собирать или чинить рамы ПА, а также с легкостью навешивать дополнительное оборудование. Внешний вид АНПА Poseidon представлен в Приложении Е.

Выводы по созданию АНПА Посейдон

После создания целого аппарата и проведения тестов, мы выяснили, что Посейдон имеет значительные недостатки: плохая герметичность, робот переворачивается и заваливается под водой, невозможно моментально обновить ПО, перегрев в блоке электроники, при создании автопилота требуется разработка собственного ПО для управления моторами. Учитывая вышеизложенные проблемы, было выяснено, что роботу нужны модификации. Для улучшения робота необходимо заменить блок электроники, заменить соедините проводов, перегерметизировать. Цена суммарных переделок с учетом уже затраченных ресурсов будет сравнима с покупкой готового робота, и написания автопилота для него. Проанализировав рынок АНПА было найдено

подходящее решение - АНПА MUR Middle AUV, который имеет следующие характеристики:

Характеристики аппарата Mur Middle AUV**:**
https://robocenter.net/goods/kit/middleauv/

Внешний вид Mur Middle AUV представлен в Приложении Ж.

Отличие Mur Middle AUV от Poseidon

В отличие от Посейдон, АНПА MUR Middle AUV легче на 5 кг, имеет меньшее энергопотребление, его размеры вдвое меньше, имеет 2 камеры, собственный внутренний локальный сервер, а также он имеет готовое ПО для управления моторами с помощью языка Python, что позволяет использовать технологии компьютерного зрения, при создании автопилота. После тестирования, мы приняли решение использовать MUR Middle AUV для дальнейшего создания подводного комплекса мониторинга подводно устьевых добывающих систем.

Программное управление роботом

Перед разработкой автопилота, мы определили движения, который должен выполнять АНПА. Робот должен выплыть с базы, двигаться вдоль нефтепровода на определенный участок, записывать видео с наложенным поверх распознаванием объектов, находящихся вблизи нефтепровода.

Встроенная библиотека Mur

Разработчик АНПА MUR Middle AUV имеет готовую библиотеку для взаимодействия с роботом с помощью языка программирования Python. Для работы необходимо создать файл (main), с расширением “.py” и запустить его, через интерфейс программы Mur IDE. При вызове библиотеки, нам открывается возможность получать данные с датчиков, а также управлять моторами:

1. auv.set_rgb_color(код цвета) - включение подсветки на аппарате
2. auv.set_motor_power(id мотора, мощность) – управление моторами
3. depth = auv.get_depth() –возвращает текущею глубины
4. yaw = auv.get_yaw() – возвращает значение курса
5. pitch = auv.get_pitch() – возвращает значение крена
6. image = auv.get_image_bottom() – возвращает видео с донной камеры
7. image = auv.get_image_front() – возвращает видео с передней камеры

Функция автоматического удерживания глубины

Чтобы передвигаться вдоль нефтепровода необходимо поддерживать одинаковую глубину, параллельно с выполнением других действий. Для поддержания глубины используем пропорционально дифференциальный регулятор, а именно мы замеряем текущую глубину и находим ошибку между нужной глубиной и текущей, далее умножаем ошибку на коэффициент (подбираемый экспериментально) и подаем это значение на вертикальные моторы, используя уже реализованные функции аппарата MUR. Коэффициент домножения ошибки был выявлен экспериментально при тестировании в симуляторе.

Код 1 - Функция автоматического удерживания глубины

```
def keep_depth(depth_to_set):
    try:
        error = auv.get_depth() - depth_to_set
        output = keep_depth.regulator.process(error)
        output = clamp(output, -100, 100)
        auv.set_motor_power(2, output)
        auv.set_motor_power(3, output)
```

```
    except AttributeError:
        keep_depth.regulator = PD()
        keep_depth.regulator.set_p_gain(80)
        keep_depth.regulator.set_d_gain(50)
```

Функция автоматического поддержания курса, функция разворота

При выполнении алгоритмов распознавания линии движения, требуется поворачивать на определенный, заданный градус и поддерживать его длительное время. Благодаря библиотеке MUR, мы можем получать текущее значение курса в градусах. Для поворота на заданный градус и поддержания одного курса, мы замеряем текущую глубину и находим ошибку между требуемым курсом и текущим, далее умножаем ошибку на коэффициент и подаем это значение на горизонтальные моторы, используя уже реализованные функции аппарата MUR.

Код 2 - Функция автоматического удерживания курса

```
def keep_yaw(yaw_to_set):
    def clamp_to_180(angle):
        if angle >= 180.0:
            return angle - 360.0
        if angle < -180.0:
            return angle + 360.0
        return angle
    try:
        error = auv.get_yaw() - yaw_to_set
        error = clamp_to_180(error)
        output = keep_yaw.regulator.process(error)
        output = clamp(output, -100, 100)
```

```
        auv.set_motor_power(0, -output)
        auv.set_motor_power(1, output)

    except AttributeError:
        keep_yaw.regulator = PD()
        keep_yaw.regulator.set_p_gain(0.8)
        keep_yaw.regulator.set_d_gain(0.5)
```

Функция определения направления стрелки

На вход подается изображение стрелки. Нужно определить, куда направлена стрелка, и проплыть по ее направлению. Для этого мы, исходя из цвета стрелки, делаем маску с помощью функции cv2.inRange(). На полученной маске выделяем все контуры и ищем максимальный по площади. Он и будет нашей стрелкой. Затем мы его аппроксимируем и получаем 4 отрезка находим 2 самых длинных, их общая точка – это и есть вершина стрелки. Смотрим как она лежит относительно центра изображения и движемся в правильную сторону. Пример получившегося результата представлен в Приложении К.

Функция нахождения линии и определения ее направления

Также создаем маску по цвету и находим на ней максимальный по площади контур. С помощью функции cv2.minAreRect() находим координаты прямоугольника наименьшей площади, в который можно вписать данный контур. То есть находим координаты вершин нашей линии. Находим наибольшую сторону и угол между ней и осью OX (за ось OX берется нижняя прямая изображения). Зная угол, поворачиваемся, пока угол не станет равен 90. Мы нашли направление нашей линии, теперь просто плывем прямо.

Код 3 - Движение вдоль линии

```
while True:
    sleep(0.03)
    frame = cam.read()[1]
    frame2= cv2.resize(frame,(frame.shape[1]//scale,frame.shape[0]//scale))
    frameHSV = cv2.blur(cv2.cvtColor(frame2,cv2.COLOR_BGR2HSV),(4,4))
    frameBW = cv2.inRange(frameHSV,(48, 0, 0),(180, 255, 255))
    left = -1
    rigth = -1
    i = 0
    for y in frameBW[0]:
        if y == 0 and left == -1:
            left = i
        elif y == 255 and left != -1 and rigth == -1:
            rigth = i
        i += 1
    if left <= 159 and rigth >= 159:
        auv.set_motor_power(2, -10)
        auv.set_motor_power(1, -10)
    elif left > 159:
        auv.set_motor_power(1, 0)
        auv.set_motor_power(2, -20)
    elif rigth < 159:
        auv.set_motor_power(1, -20)
        auv.set_motor_power(2, -0)
```

Функция распознавания цифры и объектов

В данном случае есть два способа решения данной задачи.

Первый способ - это использование ResNet, обученную на датасете MNIST (объёмная база данных образцов рукописного написания цифр).

Единственный минус этого способа — это то, что ResNet – классификатор, то есть она не способна самостоятельно найти цифру. Для решения этой проблемы можно использовать маску по черному цвету и прямоугольник минимальной площади для контура максимальной площади. То есть мы, как бы находим область с цифрой и на вход подаем только ее. Но у нас может быть и не один темный участок на изображении.

Второй способ – использование YOLO, обученную уже на собственном датасете. YOLO разработана для задач детекции, поэтому нам уже не придется собственноручно выделять область на изображении.

Минусы этого способа: у нас никак не получится собрать такой же большой датасет, как MNIST, поэтому может сильно страдать точность определения. Также из-за своей сложности YOLO более требовательна к характеристикам компьютера, поэтому при ее использовании бортовой компьютер может начать зависать. Код YOLO представлен в Приложении Л.

Распознавание опасных объектов поверх записи видео

По поставленному заданию, при передвижение вдоль нефтепровода необходимо распознавание лишних объектов (не нефтепровод и его опоры). Воспользуемся свёрточной нейронной сетью Yolov4. Программа должна найти на изображение не привычные ей предметы. На глубине, при освещение фарами робота, классификация по цвету не будет работать, поэтому нужно обучить систему распознавать темные фигуры, в разных местах, но с одинаковым фоном.

Сборка дата сета, будет состоять из создания фотографий, сделанных на фоне нефтепровода, с разным количеством темных объектов. После созданием воспользуемся библиотекой для python labeimg и создадим дата сет по размеченным данным. Пример получившегося результата представлен в Приложении М.

Распознавание ARUCO маркеров для навигации

АНПА должен уметь находить нужный участок на нефтепроводе. Для этого существует несколько вариантов. 1)использование пингеров, для этого необходимо расположить их и замерять расстояние до каждого. По расстоянию до каждого пингера, узнаём, на каком участке находится робот. 2) использование Aruco маркеров: через равное расстояние прикрепить Aruco маркер на нефтепровод, и при распознавании id маркера, определять проходимый участок. Для этого необходимо создать словарь Aruco, с помощью библиотеки OpenCV. Код представлен в Приложении Н.

Подготовка к тестированию

При тестирование нашей системы мы не можем использовать настоящий нефтепровод. Поэтому в качестве имитированного нефтепровода будем использовать длинные полоски желтого цвета, которые будут использоваться в бассейне, в качестве опасных объектов будем помещать подручные предметы. В программе Urho3D, мы создали 3D сцену, что схожа с реальным бассейном для тестирования. Эта сцена запускается в симуляторе MUR. Пример сцены представлен в Приложении П.

Симулятор для тестирования

АНПА MUR дает возможность управлять им в симуляторе. Можно самостоятельно создать сцену для бассейна, воссоздать условия миссии, и проверять работу программы в симуляторе. Мы соединили написанные нами функции распознавания линии, удержания глубины, угла поворота, а также

распознавания объектов в одну программу, и заставили имитированного робота передвинуть вдоль желтого нефтепровода из нашей созданной 3D сцены.

Тестирование автопилота

Запуск робота был запланирован следующими этапами:

- Каждый алгоритм в программе тестировался в адаптированном под АНПА MUR Middle Auv симуляторе.

- Погружение робота в бассейн, и проверка удержания курса и глубины. Робот удерживал глубину 40 см и курс 180 градусов.

- Установили на дно имитированный нефте-газопровод, вдоль которого двигался робот, также робот записывал видео, с передвижением на заданным участком пути. Полученный результат можно увидеть в этом видео. https://drive.google.com/file/d/1O_obmSPBVCbMOttAr_iMDQFv2LBttY_O/view?usp=sharing

ЗАКЛЮЧЕНИЕ

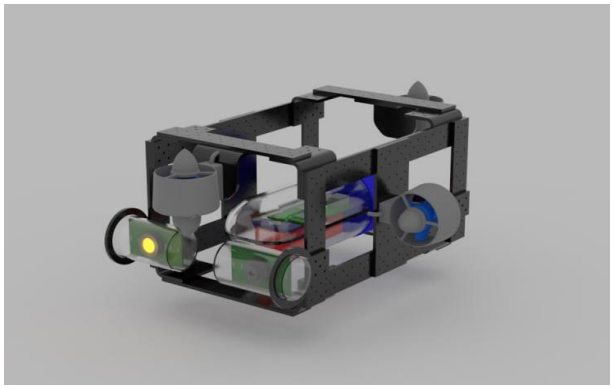
В процессе работы над своим проектом, мы реализовали все поставленные задачи. Мы создали автопилот для малогабаритного подводного автономного аппарата для мониторинга подводных систем**.** АНПА Middle AUV, с созданным нами автопилотом передвигался вдоль имитированного нефтепровода, а также производил запись видео с выделением цветом опасных объектов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Анатолий Корендясев, Теоретические основы робототехники - М.: Наука, 2006 г. - 376 с.
2. URL: https://neftegaz.ru/news/dobycha/520947-v-dvfu-razrabatyvayut-apparatdlya-kontrolya-za-sostoyaniem-podvodnykh-dobychnykh-platform/
3. Изд-во ДВО РАН, Автономные подводные роботы. Системы и технологии. Электронный учебник. – 398 с.
4. Норенков И. П. Автоматизированное проектирование: Учеб. для вузов. 2-е изд., перераб. и доп. - М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. - 336 с.
5. URL: http://www.tetis-pro.ru/faq/8028/
6. О.М. Киселёв Математические основы роботехники – Орёл: Издательство «Картуш», 2019. – 228 с.
7. URL: https://www.ge.com/ru/content/подводные-системы-для-добычи-нефти
8. URL: https://neftegaz.ru/science/booty/331858-innovatsionnye-tekhnologiipodvodnoy-dobychi-uglevodorodov-na-shelfe-arktiki/
9. URL: https://www.ge.com/ru/content
10. URL: http://vestigas.ru/sites/default/files/attachments/problemy_sozdaniya_05.pdf

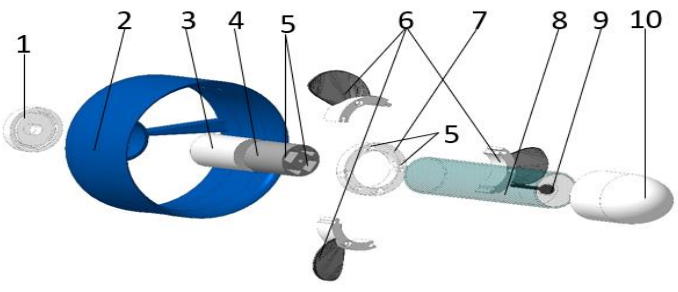
ПРИЛОЖЕНИЯ

Приложение А
3Д модель аппарата “Poseidon”



Приложение Б
Конструктор для создания корпуса

Приложение В
Конструкция движителя

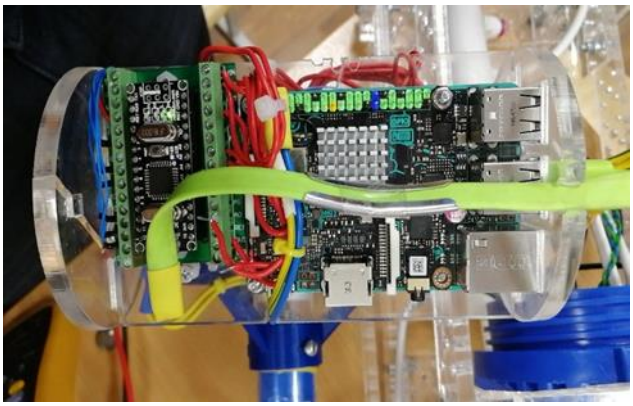


Конструкция движителя

- 1 – Задняя заглушка цилиндра
- 2 – Боковой обтекатель движителя
- 3 – Электромотор
- 4 – Внутренняя магнитная полумуфта
- 5 – Место для постоянных магнитов
- 6 – Лопасти
- 7 – Магнитное кольцо
- 8 – Цилиндр
- 9 – Передняя заглушка с центровочной осью
- 10 – Передний обтекатель

Приложение Г
Мотор Truster 300M

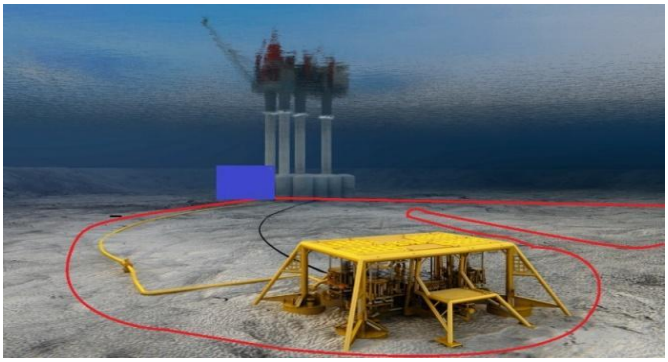
Приложение Д
Блок электроники

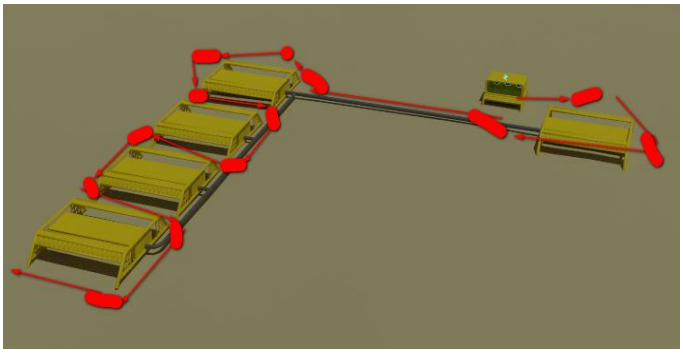


Приложение Е
Внешний вид аппарата “Poseidon”

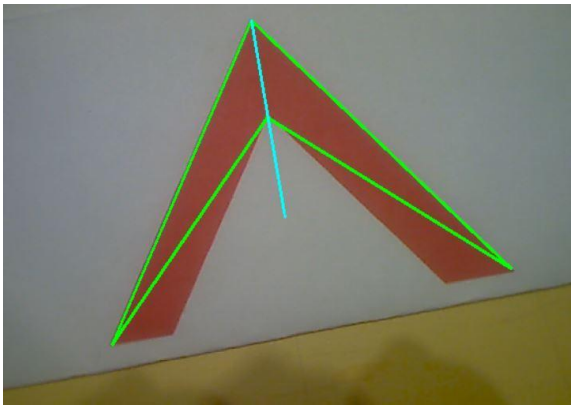
Приложение Ж
Внешний вид аппарата “Mur Middle AUV”

Приложение И
Схема перемещения вдоль нефтепровода





Приложение К
Распознавание направления стрелки



Приложение Л
Код YOLO

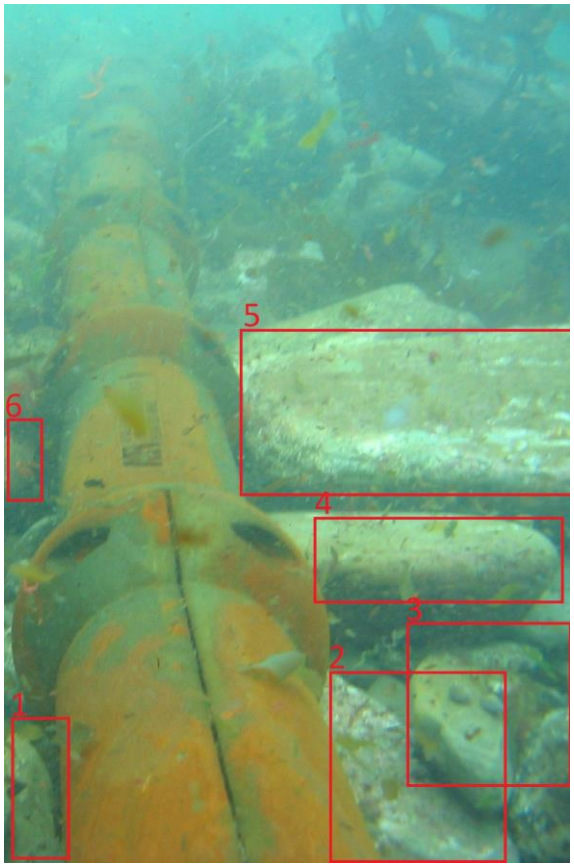
```
import cv2
import numpy as np
def get_output_layers(net):
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    return output_layers
def draw_prediction(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    label = str(classes[class_id])
    color = COLORS[class_id]
    cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
    cv2.putText(img, label, (x-10,y-
10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
classes = None
with open('yolov3.txt', 'r') as f:
    classes = [line.strip() for line in f.readlines()]
```

```
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
net = cv2.dnn.readNet('yolov3.weights','yolov3.cfg')
image = cv2.imread('1.jpg')
Width = image.shape[1]
Height = image.shape[0]
scale = 0.00392
blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)
net.setInput(blob)
outs = net.forward(get_output_layers(net))
class_ids = []
confidences = []
boxes = []
conf_threshold = 0.5
nms_threshold = 0.4
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            center_x = int(detection[0] * Width)
            center_y = int(detection[1] * Height)
            w = int(detection[2] * Width)
            h = int(detection[3] * Height)
            x = center_x - w / 2
            y = center_y - h / 2
            class_ids.append(class_id)
            confidences.append(float(confidence))
            boxes.append([x, y, w, h])
```

```
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
for i in indices:
    i = i[0]
    box = boxes[i]
    x = box[0]
    y = box[1]
    w = box[2]
    h = box[3]
    draw_prediction(image, class_ids[i], confidences[i], round(x), round(y), round(x+
w), round(y+h))
cv2.imshow("object detection", image)
```

Приложение М

Получившийся результат при распознавании опасных объектов



Приложение Н
Код распознавания ARUCO маркеров

```
import cv2
from matplotlib import pyplot as plt
def show_img_with_matplotlib(color_img, title, pos):
    # Convert BGR image to RGB
    img_RGB = color_img[:, :, ::-1]

    ax = plt.subplot(1, 3, pos)
    plt.imshow(img_RGB)
    plt.title(title)
    plt.axis('off')
fig = plt.figure(figsize=(12, 5))
plt.suptitle("Aruco markers creation", fontsize=14, fontweight='bold')
fig.patch.set_facecolor('silver')
aruco_dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
aruco_marker_1 = cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2, sidePix
els=600, borderBits=1)
aruco_marker_2 = cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2, sidePix
els=600, borderBits=2)
aruco_marker_3 = cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2, sidePix
els=600, borderBits=3)
cv2.imwrite("marker_DICT_7X7_250_600_1.png", aruco_marker_1)
cv2.imwrite("marker_DICT_7X7_250_600_2.png", aruco_marker_2)
cv2.imwrite("marker_DICT_7X7_250_600_3.png", aruco_marker_3)
show_img_with_matplotlib(cv2.cvtColor(aruco_marker_1, cv2.COLOR_GRAY2BG
R), "marker_DICT_7X7_250_600_1", 1)
```

```
show_img_with_matplotlib(cv2.cvtColor(aruco_marker_2, cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_2", 2)
show_img_with_matplotlib(cv2.cvtColor(aruco_marker_3, cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_3", 3)
plt.show()
import cv2

aruco_dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
parameters = cv2.aruco.DetectorParameters_create()
capture = cv2.VideoCapture(0)
while True:
    ret, frame = capture.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    corners, ids, rejected_corners = cv2.aruco.detectMarkers(gray_frame, aruco_dictionary, parameters=parameters)
    frame = cv2.aruco.drawDetectedMarkers(image=frame, corners=corners, ids=ids, borderColor=(0, 255, 0))
    frame = cv2.aruco.drawDetectedMarkers(image=frame, corners=rejected_corners, borderColor=(0, 0, 255))
    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release everything:
capture.release()
cv2.destroyAllWindows()
```

Приложение П
Скриншот работы в симуляторе