Проектная работа

Тема: «Беспилотный аппарат по обеспечению подводной безопасности»

Выполнил:

Наконечный Дмитрий Игоревич

Ученик 9 класса

г. Севастополь
2023

Содержание:

Введение

Ещё в конце прошлого века беспилотные аппараты и ПО для них разрабатывались только в особых центрах или по государственному заказу. В XXI веке они получили широкое распространение и усовершенствовались – теперь они могут решать широкий круг задач почти в любой отрасли.

То, как беспилотник будет использован, лишь незначительно меняет его структуру и логику его работы (если при этом не меняется внешняя среда). У каждого беспилотника есть типичное техническое оснащение (корпус, моторы, камера) и ПО, например алгоритмы управления моторами и передвижения в пространстве, анализа окружающего пространства.

Я решил сделать главной задачей программируемого мной беспилотника обеспечение безопасности под водой (как в мирное время, например мониторинг состояния аквалангиста-туриста во время дайвинга, так и в военный период – обнаружение подозрительных объектов)

Данный проект направлен на создание алгоритмов управления беспилотниками и распознавания объектов под водой, в том числе жестов аквалангиста.

Актуальность

Данный беспилотник и программное обеспечение к нему можно будет адаптировать для большого количества прикладных задач, в том числе коммерческих – от исследования морского дна до отслеживания туристов-аквалангистов во время дайвинга.

Направление: Информатика (программирование), Беспилотный транспорт и логистические системы.

Цель: создать алгоритмы перемещения подводного аппарата и распознавания объектов под водой.

Задачи:

1. Изучить спектр технологий, которые могут быть использованы для решения задачи и выбрать подходящие.
2. Спланировать архитектуру кода
3. Написать код, реализующий нужные алгоритмы
4. Протестировать код в симуляторе
5. Найти способ устранить ошибки, если они есть, то перейти к задаче номер 3.

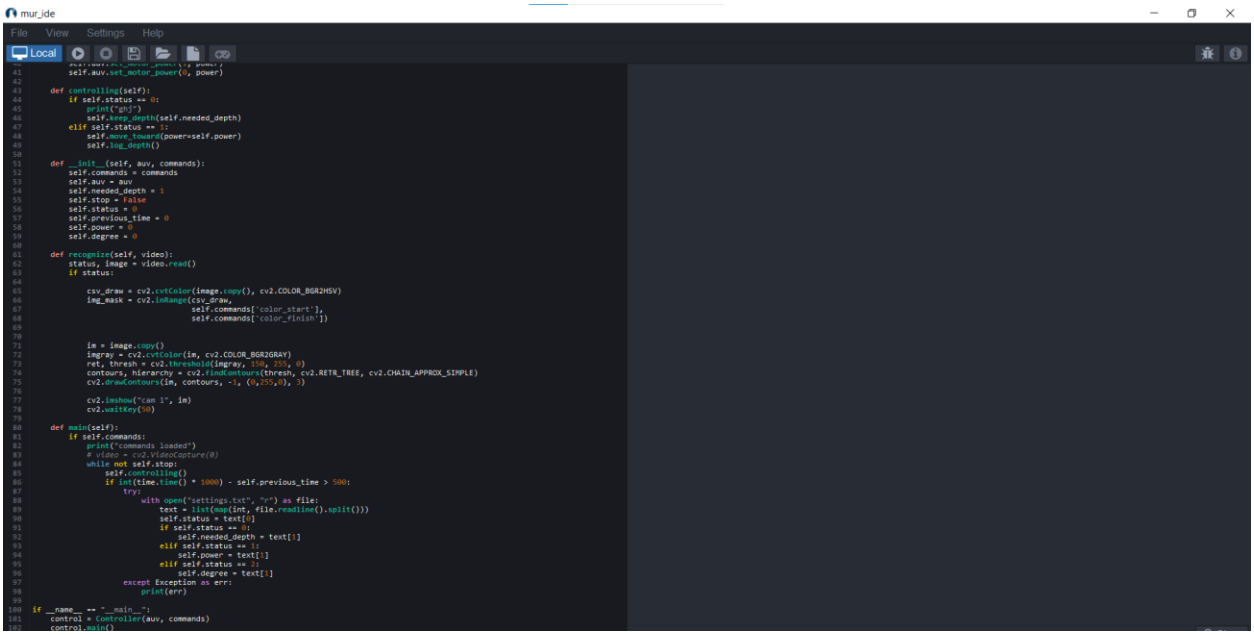
Теоретическая часть

Подводный аппарат MiddleAUV

Этот подводный беспилотник имеет 4 мотора, способные работать в разных режимах и с регулируемой мощностью. Расположены по углам аппарата, за счёт чего обеспечивают максимально возможный момент силы и манёвренность за счёт этого. Также на нём установлены 2 камеры – фронтальная и нижняя. Может передвигаться только вперед и назад, вверх и вниз относительно собственного нуль-азимута: для поворота используется включение одного из задних двигателей.

Среда разработки MurIDE

Данная среда разработки позволяет управлять подводным аппаратом используя язык программирования Python версии 3.5. В ней есть встроенный симулятор, написанный на основе Visual C++.



Язык программирования Python

Python – невероятно обширный язык программирования высокого уровня с удобным синтаксисом, использование которого направлено на повышение эффективности разработчика. Он поддерживает множество парадигм программирования несмотря на то, что основной из них является объектно-ориентированная (она опирается на принцип, гласящий, что всё является объектом определённого типа, а для каждого типа определён обособленный класс с его уникальными свойствами). Его создал Гвидо ван Россум в 1991 году. Он удобен в использовании, имеет встроенные библиотеки для решения различных задач.

OpenCV – библиотека машинного зрения

OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Обертку для этой библиотеки можно установить и для python. Оно позволяет не только, например, распознавать контуры на изображениях, но и выполнять с изображениями почти любые преобразования с ними, представляя их в качестве трёхмерной матрицы

Практическая часть

Организация структуры проекта

Основные шаги, которые я выполнял перед написанием кода:

1. Установка MurIDE
2. Установка библиотеки OpenCV
3. Настройка виртуального окружения и дополнительных файлов в проекте
4. Подготовка симулятора к работе (установка шаблона xml)

Написание кода

Код будет почти полностью написан на Python, поэтому файлы будут иметь расширение .py .

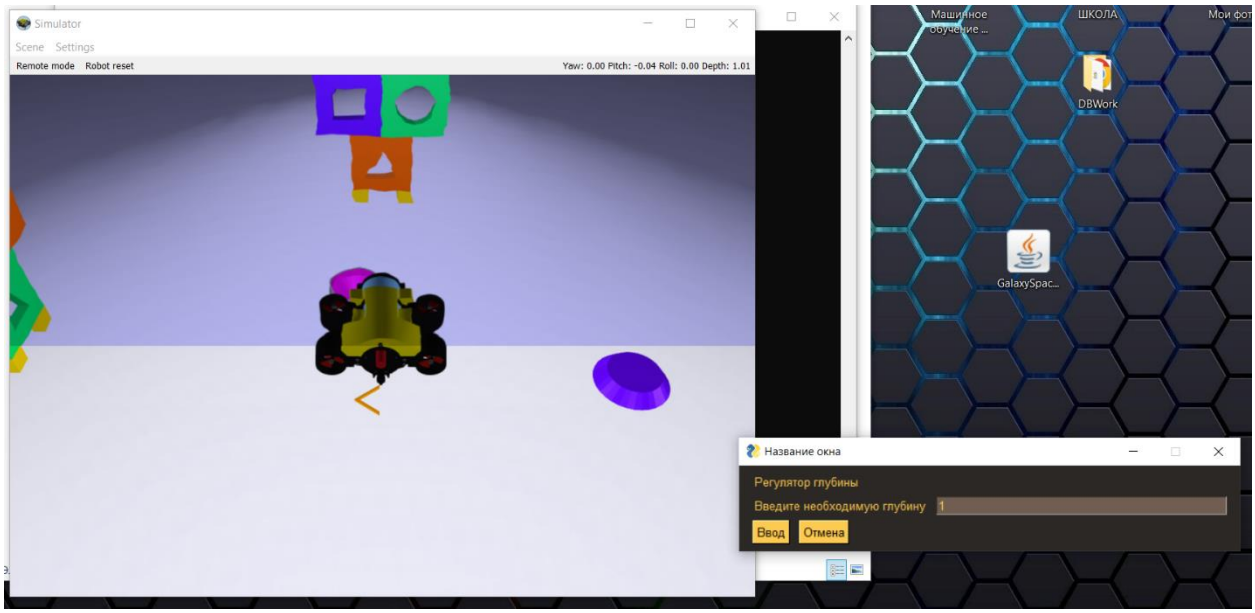
```
import pymurapi as mur
import cv2
import time

def clamp(v, min, max):
if v < min:
return min
if v > max:
return max
return v

commands = {"ignore": 15, "recognize": True,
"color_start": (0, 0, 0),
"color_finish": (27, 255, 255)}

auv = mur.mur_init()

class Controller():
def log_depth(self):
print(["зелёный цвет", "синий цвет", "фиолетовый цвет"][int(self.auv.get_depth() > 3)])
def keep_depth(self, depth_to_set):
cur_depth = self.auv.get_depth()
```

```
motor = 80 * (cur_depth - depth_to_set)
self.auv.set_motor_power(3, clamp(int(motor), -100, 100))
self.auv.set_motor_power(2, clamp(int(motor), -100, 100))
color_rgb = (0, 0, 0)
# if cur_depth > 3: self.flag = False
self.log_depth()
def swap_yaw(self, degree=0):
yaw = self.auv.get_yaw()
recalculated_yaw = yaw if (0 <= yaw <= 180) else 360 + yaw
motor = 70 * (recalculated_yaw - degree)
self.auv.set_motor_power(1, clamp(int(motor), -100, 100))
print("povorot")

def move_toward(self, power=0):
self.auv.set_motor_power(1, power)
self.auv.set_motor_power(0, power)
def controlling(self):
if self.status == 0:
print("ghj")
self.keep_depth(self.needed_depth)
elif self.status == 1:
self.move_toward(power=self.power)
self.log_depth()
def __init__(self, auv, commands):
self.commands = commands
self.auv = auv
self.needed_depth = 1
self.stop = False
self.status = 0
self.previous_time = 0
self.power = 0
self.degree = 0
def recognize(self, video):
status, image = video.read()
if status:
csv_draw = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2HSV)
img_mask = cv2.inRange(csv_draw,
self.commands['color_start'],
self.commands['color_finish'])
im = image.copy()
imgray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(imgray, 150, 255, 0)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(im, contours, -1, (0,255,0), 3)
cv2.imshow("cam 1", im)
cv2.waitKey(50)
def main(self):
if self.commands:
print("commands loaded")
# video = cv2.VideoCapture(0)
while not self.stop:
self.controlling()
```

```
if int(time.time() * 1000) - self.previous_time > 500:
try:
with open("settings.txt", "r") as file:
text = list(map(int, file.readline().split()))
self.status = text[0]
if self.status == 0:
self.needed_depth = text[1]
elif self.status == 1:
self.power = text[1]
elif self.status == 2:
self.degree = text[1]
except Exception as err:
print(err)

if __name__ == "__main__":
control = Controller(auv, commands)
control.main()
        print("task completed")
```

Запуск проекта

Проект можно запустить на самом аппарате или в симуляторе. Я отобразил результаты моей работы в виде скриншотов из симулятора.



Здесь вы видите программу «регулятор глубины» и сам симулятор

Список использованной литературы:

Бэрри П., Изучаем программирование на Python, пер. с англ. М. А. Райтман, Эксмо, 2019, 624 с.

https://opencv.org/

https://murproject.com/