Летающая робототехника

2022/23 учебный год

Второй отборочный этап

Задача IV.1. Сборка квадрокоптера (8 баллов)

Темы: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все электронные компоненты, которые были подключены и установлены неправильно.

Ссылка на видео: https://www.youtube.com/watch?v=U8Esuf0hMIw.

1. подключение шлейфа радиоприемника к полетному контроллеру;
2. подключение шлейфа радиоприемника к радиоприемнику;
3. подключение шлейфа питания к плате распределения питания;
4. подключение шлейфа питания к полетному контроллеру;
5. подключение сигнального провода регуляторов оборота от мотора №1 к полетному контроллеру;
6. подключение сигнального провода регуляторов оборота от мотора №2 к полетному контроллеру;
7. подключение сигнального провода регуляторов оборота от мотора №3 к полетному контроллеру;
8. подключение сигнального провода регуляторов оборота от мотора №4 к полетному контроллеру;
9. подключения силовых проводов от регулятора оборотов мотора №1 к плате распределения питания;
10. подключения силовых проводов от регулятора оборотов мотора №2 к плате распределения питания;
11. подключения силовых проводов от регулятора оборотов мотора №3 к плате распределения питания;
12. подключения силовых проводов от регулятора оборотов мотора №4 к плате распределения питания;
13. подключение питания светодиодной ленты;
14. подключение сигнального провода светодиодной ленты к `Raspberry PI`;
15. подключение питания (5V) к `Raspberry Pi`;
16. подключение лазерного дальномера к `Raspberry Pi`;
17. подключение шлейфа от камеры к `Raspberry Pi`;
18. подключение проводной связи от полетного контроллера к `Raspberry Pi`;
19. установка пропеллера на мотор №1;
20. установка пропеллера на мотор №2;
21. установка пропеллера на мотор №3;
22. установка пропеллера на мотор №4.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/769532/step/7?unit=771988.

Или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.html.

Ответ: 1, 5, 6, 8, 15, 17, 18, 19, 20, 21.

Задача IV.2. Настройка квадрокоптера (5 баллов)

Темы: базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все ошибки, допущенные при настройке квадрокоптера.

Ссылка на видео: https://www.youtube.com/watch?v=WKVFWyg_WQs.

1. ошибка загрузки прошивки в полетный контроллер;
2. ошибка в выборе конфигурации рамы квадрокоптера;
3. ошибка при калибровке компаса;
4. ошибка при калибровке гироскопа;
5. ошибка при калибровке акселерометра;
6. ошибка при калибровке уровня горизонта;
7. ошибка при установке ориентации полетного контроллера;
8. ошибка при калибровке радиоаппаратуры управления;
9. ошибка при настройке режимов полетного контроллера;
10. ошибка при назначении аварийного отключения моторов;
11. ошибка при настройке параметров питания;
12. ошибки при калибровке регуляторов (ESC).

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/769532/step/7?unit=771988.

Или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.html.

Ответ: 7, 10, 11.

Задача IV.3. Датчик дыма (8 баллов)

Темы: 3D-моделирование.

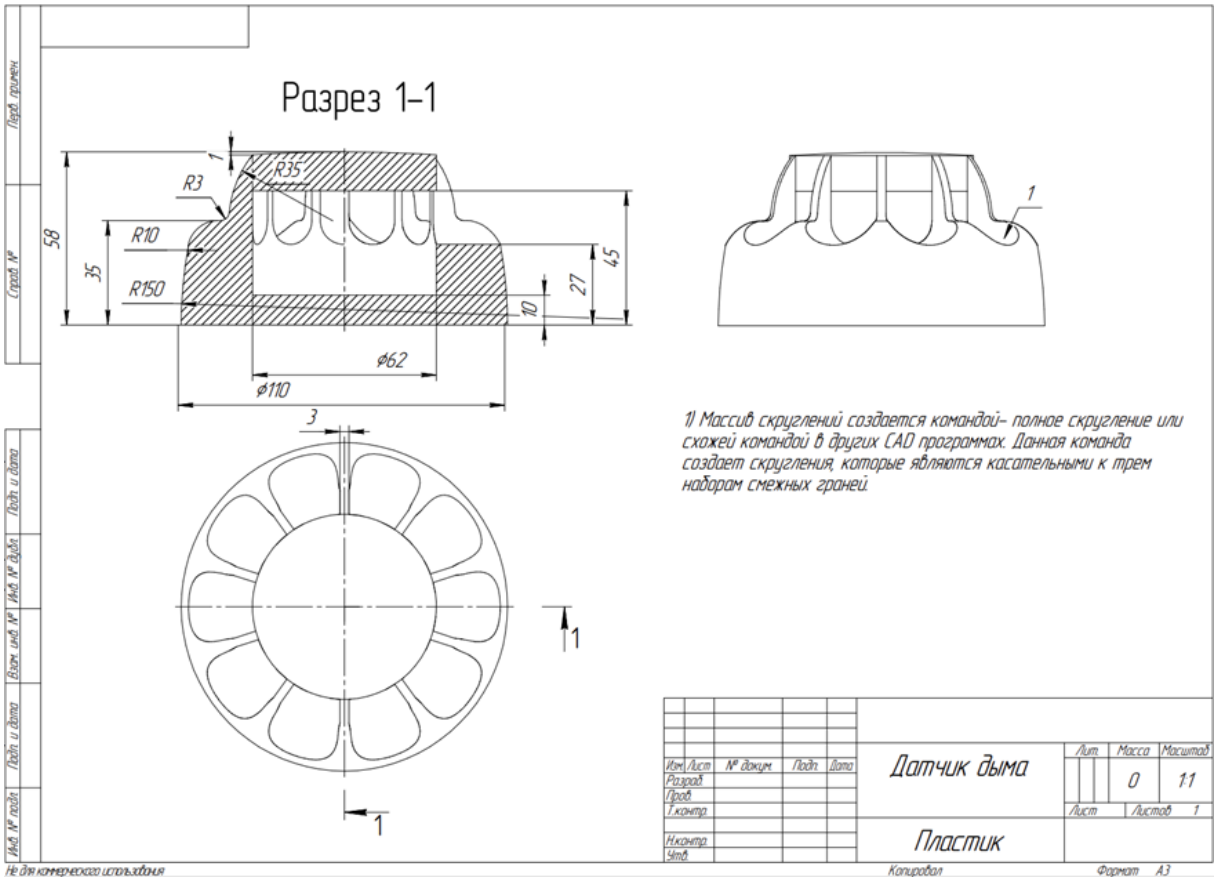
Условие

Борьба с пожарами должна быть комплексной!

Квадрокоптер используются для обнаружения источников пожара уже после его возникновения, а для того, чтобы локализовать пожар в момент его зарождения используются датчики дыма.

Постройте 3D модель датчика дыма по имеющемуся чертежу.

Материал: Пластик с плотность 1200 кг/м^3.



Примечание: необходимо использовать программу `SolidWorks` 2020 г. или `Компас-3D V20`.

В ответ запишите массу детали в граммах.

Ответ округлите до сотых, формат записи может быть любым: 75,16 или 75.16.

Решение

Для решения задачи необходимо создать 3D-модель датчика в программе `SolidWorks`, согласно представленному в тексте задачи.

Ответ: 305.

Задача IV.4. Квадрокоптеры-пожарные (10 баллов)

Темы: базовые навыки работы с летающими робототехническими системами.

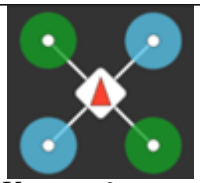
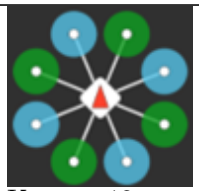
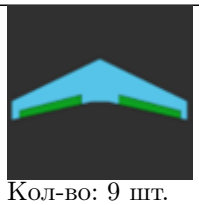
Условие

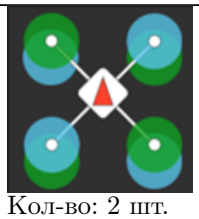
Вашей компании пришел срочный заказ на изготовление 6 пожарных БПЛА которые должны выполнять задачи по тушению локальных участков пожара. В связи со срочностью заказа вы можете использовать только компоненты имеющиеся на складе. Нужно подобрать оптимальные компоненты чтобы получившиеся БПЛА могли обработать максимальный участок пожара. Чтобы выполнить это условие БПЛА должен иметь возможность поднять большой объем воды и расходовать ее за один полет, также должна быть предусмотрена возможность его позиционирования по видеопотоку и зависания на месте. ВМГ считается оптимально подобранной в случае, если беспилотник может зависать при уровне газа не более 50%.

Постоянный расход воды 1,4 л/м.

Массу беспилотника принимать как — масса АКБ + масса бака с водой.

Список компонентов.

№	Рама	Мотор	PROP	ESC	LIPO	Полетный контроллер	Бак
A	 Кол-во: 6 шт.	Tmotor Antigravity MN4004 KV300 Кол-во: 73 шт.	Tmotor G30×10,5 Кол-во: 100 шт.	Hobbywing SKYWALKER 12a 2s Кол-во: 57 шт.	24S36P LIION 21700 4500mah Масса: 60 кг Кол-во: 17 шт.	MAMBA MK4 F722 MINI Кол-во: 8 шт.	Обьем: 100 л
B	 Кол-во: 3 шт.	Tmotor U13II KV65 Кол-во: 52 шт.	Tmotor NS24×7,2-10mm Hole Кол-во: 75 шт.	Tmotor FLAME 200A 14S Кол-во: 4 шт.	CNHL Black Series 6S 22,2V 1500mAh 100C Lipo Масса: 254 гр. Кол-во: 400 шт.	Pixracer R15 Кол-во: 21 шт.	Обьем: 20 л
C	 Кол-во: 12 шт.	Karearea TOA Bi-Turbine 2507-1450KV Кол-во: 21 шт.	Dalprop New Cyclone T5143,5 V2 Кол-во: 1000 шт.	Dual FSESC4,20 100A Кол-во: 25 шт.	24S16P LIION 21700 4500mah Масса: 28 кг Кол-во: 9 шт.	Matek F411-WTE Кол-во: 12 шт.	Обьем: 50 л
D	 Кол-во: 9 шт.	Tmotor U15II KV100 Кол-во: 25 шт.	Tmotor V40×16 Кол-во: 80 шт.	Tmotor ALPHA 80A 24S Кол-во: 48 шт.	ONBO 5200mAh 6S 50C Lipo Pack Масса: 800гр Кол-во: 20 шт.	ZMR NX4 PRO EVO Кол-во: 33 шт.	Обьем: 1000 л

№	Рама	Мотор	PROP	ESC	LIPO	Полетный контроллер	Бак
E	 Кол-во: 2 шт.	Tmotor V602 KV180 Кол-во: 85 шт.	Gemfan LR 7035–2 Кол-во: 30 шт.	Tmotor ALPHA 60A 6S Кол-во: 42 шт.	12S2P LIION 21700 4500mah Масса: 2 кг Кол-во: 11 шт.	HEX CUBE ORANGE (ADS-B Carrier Board) Кол-во: 7 шт.	Обьем: 5 л

В ответе запишите последовательность из 7 букв, порядок которых должен соответствовать столбцам таблицы, пример: ACDAEEA.

Решение

1. Выбираем раму.
   Под условия задачи не подходят варианты B, D, так как они не имеют возможности зависать на месте, и вариант E, так как количества рам недостаточно. Возможные варианты A, C.
2. Поскольку необходимо, чтобы дрон поднимал большое количество воды, то варианты двигателей A, C и E не подходят. Остаются варианты B и D, но D совместим только с рамой типа A.
3. Заходим на сайт производителя моторов и выбираем пропеллеры, рекомендованные производителем. Для двигателя B — это Prop A. Для двигателя D — это D. Остальные пропеллеры имеют слишком малый диаметр и не подходят.
4. Варианты ESC A и E не подходят ни для одного из выбранных моторов. Вариант C не подходит, т. к. это регуляторы для электротранспорта. Вариант B — в недостаточном количестве. Правильный ответ — D.
5. Рассмотрим 3 варианта винтомоторной группы:
   5.1. Вариант 1. Мотор D, рама A, prop D. Исходя из таблицы испытаний на сайте производителя, тяга ВМГ при 50% газа равна 51,2 кг, а ток висения примерно равен 125,2A. Из этого следует, что наибольший бак, который можно использовать — B и аккумулятор C.
   5.2. Вариант 2. Мотор B, рама C, prop A. Исходя из таблицы испытаний на сайте производителя, тяга ВМГ при 50% газа равна 78,1 кг, а ток висения примерно равен 115,2A. Из этого следует, что наибольший бак, который можно использовать — C и аккумулятор C.
   5.3. Вариант 3. Мотор B, рама A, prop A. Исходя из таблицы испытаний на сайте производителя, тяга ВМГ при 50% газа равна 39 кг, а ток висения примерно равен 57,6A. Из этого следует, что наибольший бак, который можно использовать — E и аккумулятор C.
6. Сравнив 3 варианта ВМГ, можно прийти к выводу, что наилучшим образом нам подходит вариант 2. Чтобы это проверить, сделаем приблизительный расчет времени полета.
   Общая емкость аккумулятора равна $4,5 \text{ A} \cdot 16 = 72$ A, ток висения 115,2 A. Из этого следует, что время полета приблизительно 36 минут. Так как расход воды равен 1,4 литра в минуту, то за время полета расходуется ровно 1 бак объемом 50 л.
   В итоге правильный ответ под вариантом 2.

7. Варианты LIPO A, B, C, E не подходят, так как у них слишком низкое напряжение. Варианты полетных контроллеров A, D, C не подходят, так как он не обеспечат автономный полет. Вариант B не подходит, так как он не предназначен для 8 моторной схемы беспилотника. Правильный ответ — E.

Ответ: CBADCEC.

Задача IV.5. Оптимизация маршрута 2.0 (11 баллов)

Темы: математика, физика, понимание работы с летающих робототехнических систем.

Условие

Октокоптер вылетает из пожарной станции (А) с водой к точке возгорания (В), после сброса воды октокоптер может набрать ее в водоеме (Б). Пожарная станция (А) и водоем (Б) находятся на одной прямой вдоль дороги, а точка возгорания (В) находится на расстоянии 49 м от этой дороги. До проекции от точки В со стороны точки А — 274 м, а со стороны точки Б — 99 м. Зависимость скорости от уровня газа (в процентах):

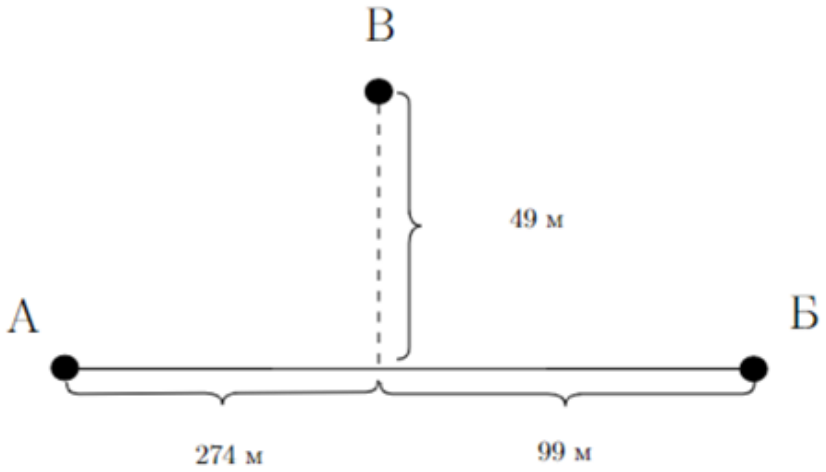
- дрон не нагружен: $v = 9 \cdot \frac{q-40}{60}$;
- дрон нагружен: $v = 4 \cdot \frac{q-40}{60}$;

где $q \in [40, 100]$ — это процент газа,

v — скорость в м/с.

- Потребление тока одним двигателем (Ампер): $\sigma = 15 \cdot \frac{q-40}{60} + 2$.
- Газ выше 80% может быть только по пути к точке возгорания, а выше 95% только над дорогой.
- Над дорогой дует ветер попутный ветер, который который увеличивает скорость дрона на 3 м/с.
- Объем аккума составляет 8 ампер-час, на момент вылета заряжен на 74%.

Сколько процентов заряда батареи осталось на момент прилета в точку Б, при условии, что октокоптер прилетел за минимальное возможное время?



Решение

Сначала рассмотрим путь из точки A в точку B.

Самая оптимальная траектория, это лететь вдоль прямой до определенного момента X, а после лететь опять по прямой только от точки X к B. Почему эта траектория оптимальная, если рассмотреть отдельно, то прямая — это самый короткий путь, тогда почему же не полететь с A сразу в B, так как вдоль прямой у нас есть ветер, который увеличивает нашу скорость, а вне ее мы можем использовать только 95% мощности (по условиям), получаем, что сразу по двум причинам выгодно сначала двигаться с большей скоростью, а потом с меньшей от точки X до B (логично, что с большей скоростью мы преодолеем большую часть пути).

Найдем с помощью теоремы Пифагора расстояние от точки X до точки B:

$$p = \sqrt{(274 - x)^2 + 49^2}$$

На отрезке A–X: $q = 100$, следовательно, $v = 4 \cdot \frac{100-60}{40} = 4$ м/с, к этому нужно еще прибавить 3 м/с (по условию).

На отрезке X–B: $q = 95 \Rightarrow v = 4 \cdot \frac{95-60}{40} = 3,5$ м/с.

Необходимо минимизировать время, вспомним формулу из школьной физики: $t = S/V$.

$$t_a(x) = \frac{x}{7} + \frac{\sqrt{(274 - x)^2 + 49^2}}{3,5}$$

Найдем производную от этой функции, получим:

$$t_a(x)' = \frac{1}{7} + \frac{2x - 548}{7\sqrt{(274 - x)^2 + 49^2}}$$

Решив уравнение $t_a(x)' = 0$, найдем экстремум (минимум) функции $t_a(x)$, равный $x = 274 - \frac{49}{\sqrt{3}} \approx 245,70984$, подставив это значение в функцию получим минимальное время, затраченное на путь A–B равное 51,26721 с.

Найдем потребление батареи на отрезке A–B одним двигателем.

На отрезке A–X октокоптер пролетел $x = 245,70984$ метров со скоростью 7 м/с, то есть потретал:

$$t_1 = \frac{245,70984}{7} \approx 35,10141 \text{ секунд времени.}$$

Один двигатель потребляет $\sigma = 15 \cdot \frac{100-40}{60} + 2 = 17$ ампер в час, то есть за час он способен посадить аккумулятор объемом 17000 mAh. Посчитаем сколько он потратит за 35,10141 секунд:

$$\frac{17000}{3600} \cdot 35,10141 = 165,75666 \text{ миллиампер.}$$

На отрезке X–B газ равен 95, а время $t_2 = t_{min} - t_1 = 51,26721 - 35,10141 =$
$= 16,1658$ с.

Найдем потребление на этом участке: $\sigma = 15 \cdot \frac{95-40}{60} + 2 = 15,75$, то есть 15750 mAh.

На этом участке один двигатель потратит $\frac{15750}{3600} \cdot 16,1658 = 70,7254$ миллиампер.

Проведя аналогичные действия с участком В–Б, учитывая, что на том участке максимальный газ равен 80, скорость рассчитывается по иной (данной) формуле, по которой возможно движение с большей скоростью (за счет уменьшения массы из-за сброшенной воды), получим потребление одним двигателем на участке В–Y (Y — точка на прямой АБ, справа от проекции точки В, в которую прилетит дрон, чтобы дальше двигаться по прямой к точке Б) равное 36,52243 mAh, а на участке Y–Б — 20,43447 mAh.

Подводим итоги.

Мы вели расчеты для одного двигателя, а у нас октокоптер, у него 8 двигателей, следовательно, всего потрачено:

$$(165,75666 + 70,7254 + 36,52243 + 20,43447) \cdot 8 = 2347,51168\text{mAh}.$$

Найдем сколько это процентов от нашего аккумулятора:

$$\frac{2347,51168}{8000} \cdot 100 \approx 29,3439.$$

Ответ: $74 - 29,3439 = 44,6561 \approx 44,7$ погрешность (± 1).

Ответ: 44,7±1.

Задача IV.6. Автономный полет. Настройка образа (6 баллов)

Темы: настройка квадрокоптера, базовые навыки работы с летающими робототехническими системами.

Условие

Ознакомьтесь с видео и отметьте все ошибки допущенные при настройке квадрокоптера.

Примечание: сборка квадрокоптера является стандартной.

1. Настройка параметра «`aruco`».
2. Настройка параметра «`aruco_detect`».
3. Настройка параметра «`aruco_map`».
4. Настройка параметра «`aruco_vpe`».
5. Настройка параметра «`map`».
6. Настройка размера `aruco` маркера по умолчанию.
7. Настройка параметра «`direction_z`».
8. Настройка параметра «`direction_y`».
9. Настройка параметров карты: длина маркера.
10. Настройка параметров карты: количество маркеров по оси X.
11. Настройка параметров карты: количество маркеров по оси Y.
12. Настройка параметров карты: расстояние между центрами меток по оси X.
13. Настройка параметров карты: расстояние между центрами меток по оси Y.
14. Настройка параметров карты: номер первого маркера.
15. Перезагрузка пакетов клевера на образе.

Решение

Для решения задачи необходимо внимательно посмотреть видео по сборке квадрокоптера из образовательного курса: https://stepik.org/lesson/769532/step/7?unit=771988.

Или ознакомиться с документацией по сборке квадрокоптера: https://clover.coex.tech/ru/assemble_4_2_ws.html.

Ответ: 1, 5, 8, 10, 11, 14.

Задача IV.7. Карта помещения (8 баллов)

Темы: программирование (Python), работа с массивами.

Условие

Постройте карту помещения, учитывая данные, получаемые с ультразвуковых датчиков квадрокоптера. Датчики установлены на всех четырех сторонах квадрокоптера. Значения получаемые с ультразвуковых датчиков имеют два вида:

- 0 — отсутствие препятствия;
- 1 — наличие препятствия.

Формат входных данных

- Первая строка (`length, width`) размер карты.
- Вторая строка — число n, количество строк далее.
- Далее идут n строк формата x, y, f, r, b, l:
  - x, y — координат где находится дрон;
  - f, r, b, l — данные с ультразвуковых датчиков спереди, справа, сзади, слева соответственно

Формат выходных данных

Карта вида `length, width`, где X — неисследованная область помещения, 1 — наличие препятствия, 0 — отсутствие препятствия.

Примечание: если квадрокоптер не исследовал какую-то область помещения, то ее необходимо обозначить на карте буквой X.

Решение

1. Необходимо создать двухмерный массив размера (`length, width`), где все значения буду X — неисследованная область помещения;
2. Необходимо считать число n-данных, в которых:
   2.1. для каждой строки, мы записываем в переменную значения x, y, f, r, b, l;
   2.2. для координаты x, y, требуется записать в массив значение 0, поскольку если квадрокоптер находится в данной точке, то препятствия там быть не может;

2.3. для каждой стороны требуется рассчитать координаты:
   - спереди $x + 1$, y;
   - справа x, $y + 1$;
   - сзади $x - 1$, y;
   - слева x, $y - 1$;

2.4. также, для каждой стороны, требуется проверить, не выходят ли координаты за границы карты, если выходят, то квадрокоптер находится на краю помещения и его грань заносить в массив не требуется;

2.4.1. необходимо проверить, было ли изменено до этого значение. Для чего необходимо проверить равно ли значение X или нет. В случае, если значение равно X, записываем в него значения с датчиков для каждой стороны;

3. После выполнения п. 1 и п. 2, необходимо вывести массив в обратном порядке, чтобы координата 0, 0 была снизу слева.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
xMap, yMap = map(int, input().split())
maps = [['X' for _ in range(yMap)] for __ in range(xMap)]
n = int(input())
for _ in range(n):
    line = list(map(int, input().split()))
    x, y = line[0:2]
    maps[x][y] = 0
    if x + 1 < xMap:
        if maps[x+1][y] == 'X':
            maps[x+1][y] = line[2]
    if y + 1 < yMap:
        if maps[x][y+1] == 'X':
            maps[x][y+1] = line[3]
    if x - 1 >= 0:
        if maps[x-1][y] == 'X':
            maps[x-1][y] = line[4]
    if y - 1 >= 0:
        if maps[x][y-1] == 'X':
            maps[x][y-1] = line[5]

for line in maps[::-1]:
    print(*line)
```

Задача IV.8. Тик-так (10 баллов)

Темы: навыки работы с компьютерным зрением (OpenCV), программирование (Python).

Условие

Одним из применений дронов в области нефтегазового дела является проверка значений аналоговых датчиков, которые не имеют связи с «внешним миром» и отображают значения на стрелочном циферблате.

Сейчас перед вами стоит подобная задача, но вам предстоит работа с двумя стрелками, а считывать значения мы будем с домашних аналоговых часов.

Необходимо с использованием алгоритма технического зрения распознать какое время изображено на часах. Гарантируется, что стрелки на их концах не будут перекрывать друг друга.

Формат входных данных

Ссылка на изображение часов (функция, позволяющая скачать изображение и импортировать в `OpenCV` приложена к заданию), время с которых необходимо распознать.

Формат выходных данных

Время в формате `hh:mm`, где `hh` — часы в 12-часовом формате ($0 \leqslant$ `hh` $\leqslant 11$); `mm` — минуты ($0 \leqslant$ `mm` $\leqslant 59$). И часы и минуты требуется вывести с лидирующими нулями (4:59 неверно, 04:59 верно).

Примеры

Пример №1

Стандартный ввод
https://stepik.org/media/attachments/course/122772/clock_88e05d50-3c38-49e4-977f-8b2bf8c62b64.jpg

Стандартный вывод
10:00

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
from math import floor
from urllib.request import urlopen
import cv2  # v. 4.5.5
import numpy as np

def get_image(url: str) -> np.ndarray:
    resp = urlopen(url)
    image = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)
    return image

def calc_angle(image: np.ndarray) -> float:
    M = cv2.moments(image)

    cX = int(M["m10"] / M["m00"]) - image.shape[1] // 2
    cY = image.shape[0] // 2 - int(M["m01"] / M["m00"])
    vector = np.array([cX, cY])
    vector = vector / np.linalg.norm(vector)
    unit_vector = np.array([0, 1])

    angle = np.arctan2(np.cross(vector, unit_vector), np.dot(vector, unit_vector))
    if angle < 0:
        angle = 2 * np.pi + angle
    return angle

MINUTE_COEF = 2 * np.pi / 60
HOUR_COEF = 2 * np.pi / 12

im = get_image(input())

hour_arrow = cv2.inRange(im, (0, 0, 200), (50, 50, 255))
minute_arrow = cv2.inRange(im, (0, 0, 0), (10, 10, 10))

mask = np.zeros_like(hour_arrow)
cv2.circle(mask, (im.shape[1] // 2, im.shape[0] // 2), 140, 255, 10)

hour = cv2.bitwise_and(hour_arrow, mask)
minute = cv2.bitwise_and(minute_arrow, mask)

hours = floor(calc_angle(hour) / HOUR_COEF)
minutes = int(round(calc_angle(minute) / MINUTE_COEF))
if minutes >= 60:
    hours += 1
```

```
    minutes = 0
print(f"{hours:02d}:{minutes:02d}")
```

Задача IV.9. Поиск возгораний (11 баллов)

Темы: программирование (Python), навыки работы с компьютерным зрением (OpenCV).

Условие

Необходимо распознать на изображении воспламенения.

Их размеры могут варьироваться:

- если ширина/длина воспламенения менее 50 пикселей, то это «small zone»;
- если ширина воспламенения более 100 пикселей, то это «big zone»;
- оставшиеся — «medium zone».

Воспламенения могут иметь три различные формы: https://disk.yandex.ru/d/gGESsnFnVToEFQ.

Изображение лежит по ссылке, имеет разрешение 1200×500 (изображение стандартное, трехканальное, формат `np.uint8`).

Посчитайте общее количество воспламенений, а также количество воспламенений различного размера.

Где N — общее количество воспламенений, X — количество «small zone» , Y — количество «medium zone», Z — количество «big zone».

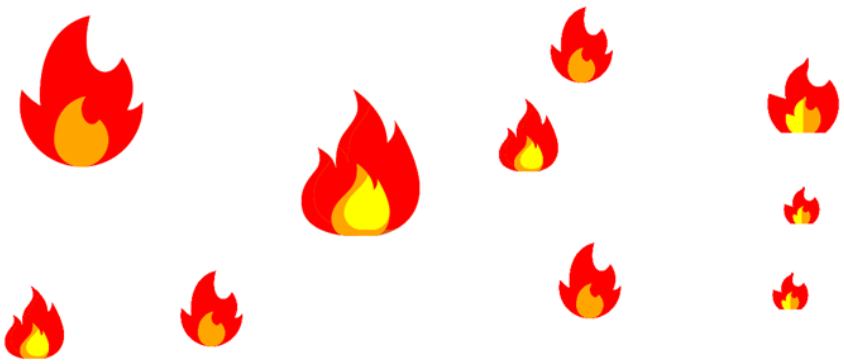


Рис. 1. Пример входного изображения

Примеры

Пример №1

Стандартный ввод
`https://stepik.org/media/attachments/course/122772/image734.png`

Стандартный вывод
`On picture detecting 12 danger zone` `6 small zone, 5 medium zone, 1 big zone`

Решение

Для решения задачи необходимо:

- наложить фильтр красного цвета (он занимает наибольшую площадь);
- найти контуры каждого возгорания;
- создать квадрат вокруг контура, произвести аппроксимацию контуров и выявить наиболее отдаленные точки;
- нанести оставшиеся цветные фильтры (желтый и оранжевый) на изображения для выявления остальных контуров;
- в случае, когда центр контура не красного цвета попадает в данный квадрат, мы суммируем их площадь;
- пройтись по всем цветам и контурам, после чего, посчитать количество контуров, имеющих площадь:
    - меньше 50×50 пикселей;
    - больше 100×100 пикселей;
    - оставшихся;
- полученные значения записываем в ответ.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import urllib.request
import numpy as np
import cv2
import random

def url_to_image(url):
    resp = urllib.request.urlopen(url)
    image = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)
    return image
def check_square(x, y, h, x1, y1, h1):
    if (x >= x1 and x <= x1+h1+h/2 and y >= y1 and y <= y1+h1+h/2) or (x+h >= x1
    ↪  and x+h <= x1+h1+h/2 and y >= y1 and y <= y1+h1+h/2) or (x >= x1 and x <=
    ↪  x1+h1+h/2 and y+h >= y1 and y+h <= y1+h1+h/2) or (x+h >= x1 and x+h <=
    ↪  x1+h1+h/2 and y+h >= y1 and y+h <= y1+h1+h/2): return True
    return False
def find_squre(approx):
    x_min = 999999999
    y_min = 999999999
    x_max = 0
    y_max = 0
    for a in approx:
        if a[0][0] > x_max:
            x_max = a[0][0]
        if a[0][0] < x_min:
            x_min = a[0][0]
        if a[0][1] > y_max:
            y_max = a[0][1]
        if a[0][1] < y_min:
            y_min = a[0][1]
    return x_min, y_min, x_max-x_min, y_max-y_min
def in_point(x, y):
```

```
    global mas
    for i in range(len(mas)):
        if check_square(x, y, 1, mas[i][0], mas[i][1], max(mas[i][2], mas[i][3])):
            return i
    return -1
img = url_to_image(input())
size = [50, 100, 200]
red_low = np.array([0, 0, 240])
red_high = np.array([1, 1, 255])

yellow_low = np.array([0, 240, 240])
yellow_high = np.array([1, 255, 255])

orange_low = np.array([0, 150, 240])
orange_high = np.array([1, 170, 255])

mask_red = cv2.inRange(img, red_low, red_high)
mask_yellow = cv2.inRange(img, yellow_low, yellow_high)
mask_orange = cv2.inRange(img, orange_low, orange_high)

mas = []
cnt = cv2.findContours(mask_red, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[1] # Поиск контуров в отфильтрованном изображении.
try:
    for c in cnt:
        moments = cv2.moments(c, 1)
        sum_pixel = moments['m00'] # Поиск количества всех пикселей необходимого цвета.
        if sum_pixel > 10:
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.01 * peri, True) # Чем меньше число тем больше шумов, чем больше тем угловатей.
            x_left, y_left, h_max, w_max = find_squre(approx)
            mas.append([x_left, y_left, h_max, w_max, sum_pixel])
except: pass

cnt = cv2.findContours(mask_yellow, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[1] # Поиск контуров в отфильтрованном изображении.
try:
    for c in cnt:
        moments = cv2.moments(c, 1)
        sum_pixel = moments['m00'] # Поиск количества всех пикселей необходимого цвета.
        y = int(moments['m10'] / sum_pixel) # Поиск координаты y центра цветного объекта.
        x = int(moments['m01'] / sum_pixel) # Поиск координаты x центра цветного объекта.
        id = in_point(x, y)
        if id != -1: mas[id][4]+=sum_pixel
except: pass

cnt = cv2.findContours(mask_orange, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[1] # Поиск контуров в отфильтрованном изображении.
try:
    for c in cnt:
        moments = cv2.moments(c, 1)
        sum_pixel = moments['m00'] # Поиск количества всех пикселей необходимого цвета.
        y = int(moments['m10'] / sum_pixel) # Поиск координаты y центра цветного объекта.
```

```
80          x = int(moments['m01'] / sum_pixel) # Поиск координаты x центра цветного
            ↪ объекта.
81          id = in_point(x, y)
82          if id != -1: mas[id][4]+=sum_pixel
83  except: pass
84  count_small = 0
85  count_medium = 0
86  count_big = 0
87  for i in mas:
88      if i[-1] < size[0]*size[0]:
89          count_small+=1
90      elif i[-1] > size[1]*size[1]:
91          count_big+=1
92      else: count_medium+=1
93  s = "On picture detecting {} danger zone".format(len(mas)) + "\n" + "{} small
    ↪ zone, {} medium zone, {} big zone".format(count_small, count_medium,
    ↪ count_big)
94  print(s)
```

Задача IV.10. Цвета объектов (7 баллов)

Темы: программирование (Python), навыки работы с компьютерным зрением (OpenCV).

Условие

Необходимо:

- распознать на изображении цветные прямоугольники и вывести количество наиболее повторяющихся (по цвету), а также информацию по наименее распространенным объектам;
- записать название наиболее распространенного цвета прямоугольников в переменную `color_max`, а наименее распространенного цвета прямоугольников — в переменную `color_min` (формат записи — на английском с маленькой буквы).

Изображение лежит в переменной `img`, имеет разрешение 2000×1000 (изображение стандартное, трехканальное, формат `np.uint8`).

Пример записи в переменную данных о цвете

```
color_max = "green"
color_min = "black"
// Список возможных цветов: 'black', 'blue', 'green', 'grey', 'orange', 'pink',
↪  'red', 'yellow'.
```

Также необходимо записать количество наиболее распространенного цвета прямоугольников в переменную `count_max`, а наименее распространенного цвета прямоугольников в переменную `count_min` (формат записи — целочисленное число).

Пример записи в переменную данных о количестве наиболее и наименее распространенных цветных прямоугольниках

```
count_max = 10
count_min = 1
```

Примечание:

- Если количество прямоугольников идентично, то необходимо выбрать цвет в соответствии с алфавитным порядком.
- Если прямоугольники накладываются друг на друга, то считать их за один.
- Версия OpenCV = 3.4.4.

Рис. 2. Пример входного изображения

Для рисунка 2 из примера соответствуют следующие значения:

```
count_min = 1
count_max = 7
color_min = "red"
color_max = "black"
```

Примеры

Пример №1

Стандартный ввод

Стандартный вывод
`All right!`

Решение

Для решения задачи необходимо:

1. Нанести на изображение цветные фильтры каждого возможного цвета для выявления прямоугольников соответствующего цвета.
2. Применить поиск контуров для каждого фильтрованного изображения и посчитать количество получившихся контуров, которое и будет являться количеством цветных прямоугольников.
3. Отсортировать полученные данные в алфавитной последовательности.
4. Отсортировать полученные данные по количеству, для вычисления наибольшего и наименьшего числа распознанных цветных прямоугольников.
5. Записать полученные значения в переменные в формате задания.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2
import numpy as np
import random

red_low = np.array([0, 0, 240])
red_high = np.array([1, 1, 255])

green_low = np.array([0, 240, 0])
green_high = np.array([1, 255, 1])

blue_low = np.array([240, 0, 0])
blue_high = np.array([255, 1, 1])

yellow_low = np.array([0, 240, 240])
yellow_high = np.array([1, 255, 255])

black_low = np.array([0, 0, 0])
black_high = np.array([1, 1, 1])

grey_low = np.array([125, 125, 125])
grey_high = np.array([130, 130, 130])

pink_low = np.array([240, 0, 240])
pink_high = np.array([255, 1, 255])

orange_low = np.array([0, 150, 240])
orange_high = np.array([1, 170, 255])

mask_red = cv2.inRange(img, red_low, red_high)
mask_green = cv2.inRange(img, green_low, green_high)
mask_blue = cv2.inRange(img, blue_low, blue_high)
mask_yellow = cv2.inRange(img, yellow_low, yellow_high)
mask_black = cv2.inRange(img, black_low, black_high)
mask_grey = cv2.inRange(img, grey_low, grey_high)
mask_pink = cv2.inRange(img, pink_low, pink_high)
mask_orange = cv2.inRange(img, orange_low, orange_high)

array = []
cv2_key = 1
array.append([len(cv2.findContours(mask_red, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'red'])
array.append([len(cv2.findContours(mask_green, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'green'])
array.append([len(cv2.findContours(mask_blue, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'blue'])
array.append([len(cv2.findContours(mask_yellow, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'yellow'])
array.append([len(cv2.findContours(mask_black, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'black'])
array.append([len(cv2.findContours(mask_grey, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'gray'])
array.append([len(cv2.findContours(mask_pink, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'pink'])
array.append([len(cv2.findContours(mask_orange, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)[cv2_key]), 'orange'])

```

```
array.sort(key=lambda array: array[1], reverse=True)
array.sort(key=lambda array: array[0])
color_max = array[-1][1]
count_max = array[-1][0]

array.sort(key=lambda array: array[1])
array.sort(key=lambda array: array[0])
color_min = array[0][1]
count_min = array[0][0]
```

Задача IV.11. Полетели (командная задача) (16 баллов)

Темы: базовые навыки работы с летающими робототехническими системами, программирование (Python), навыки работы с компьютерным зрением (OpenCV).

Условие

Группе исследователей необходимо провести мониторинг территории и собрать некоторые данные при помощи квадрокоптера. Прежде чем приступать к физическим полетам, они хотят протестировать работоспособность автономной системы при помощи симулятора Gazebo: https://clover.coex.tech/ru/simulation.html.

Датчик, установленный на квадрокоптере, может работать стабильно, когда квадрокоптер не находится в полете, и у него выключены двигатели (задизармлен). Поэтому перед полетами вам необходимо построить в стимуляторе мир, в котором необходимо установить тумбы (`Box`) на места, в которых необходимо сделать измерения. Это требуется сделать автоматически.

После этого, необходимо написать программу полета по этим точкам с посадкой на установленные тумбы и считыванием показаний с датчика.

Обратите внимание, что 9999,0 и $-1,0$ — невалидные данные датчика, означающие что что-то идет не так.

Для решения задачи вам потребуется установить виртуальную машину с симулятором `Clover`: https://clover.coex.tech/ru/simulation_vm.html или симулятор на свой компьютер с `OS Ubuntu`: https://clover.coex.tech/ru/simulation_native.html.

После этого необходимо установить имитирующее реальный датчик ПО. Для этого необходимо выполнить следующую команду в терминале:

```
wget https://gist.githubusercontent.com/bart02/12e4ef0c588d911fcef153bcca6bbc89
↪  /raw/940d1ac1f4ed266ac26a021b88a5f2e44b8ba3bb/script.sh -O - |
↪  bash
```

Часть 1. Автоматический отчет

Вам необходимо написать программу полета по этим точкам с посадкой на установленные коробки и считыванием показаний с датчика. Общий алгоритм таков:

1. Написать программный код для автоматической генерации мира `Gazebo`.
2. Подготовить программный код для выполнения квадрокоптером в сгенерированном мире следующие миссии в автономном режиме:
   2.1. Взлет с точки старта;

2.2. Полет в точку №1;
2.3. Измерение данных в точке о №1 на заданной высоте;
2.4. Полет в точку №2;
2.5. Измерение данных в точке о №2 на заданной высоте;
2.6. Полет в точку №3;
2.7. Измерение данных в точке о №3 на заданной высоте;
2.8. Полет в точку №4;
2.9. Измерение данных в точке о №4 на заданной высоте;
2.10. Полет в точку №5;
2.11. Измерение данных в точке о №5 на заданной высоте;
2.12. Вывод данных полученных данных полученных в процессе измерений в терминал.

Данная задача проверяет вывод автоматического отчета (п. 2.12) в терминал.

Формат входных данных

На вход поступают данные содержащие информация о запуске (см. пример 1) и набор точек в виде:

- команда, которую необходимо запустить в терминале для обновления имитационных данных;
- x, y, z — координаты точек в которых необходимо провести измерение некоторых данных при помощи датчиков находящихся на квадрокоптер.

Формат выходных данных

```
x1, y1, z1 = n1
x2, y2, z2 = n2
x3, y3, z3 = n3
x4, y4, z4 = n4
x5, y5, z5 = n5
```

Где x и y — координаты точек, в которых производились измерения;

z — высота, на которой производились измерения;

n — данные, полученные при измерениях.

Примеры

Пример №1

Стандартный ввод
0 Run this in the terminal: echo gAAAAABjYXIVBldkq4jIxsHtIyirDOKJzInn-ZK3VtG212jJ1YON4I8gOlw6XZ_Thkkn qHE_MbIksezDSlIKjz7TZengwRB_-BfqBeXTMMNyTZz8Skf8SoUlmrD1kO7kxZCHOWtiwyf5F JeIgkCmD_ssJADJUNU-uoIJJEROZwbcjqeA9njA1ECslmuC4ahGPm4rF9i475RpYAkb6yrdii 8fgD-fzDexMBxK6SHkkDz01Zq5Dy5TYOjr9XOiJumZEpRDJnmzBBkyFUxZFReYwC3iUcjl22u c5w== > ~/.nto/sensor.txt Then navigate the drone by these points (x, y, z): [(3.4, 7.3, 2.5), (6.7, 6.0, 2.6), (9.6, 0.2, 2.8), (9.5, 1.9, 2.1), (0.9, 9.2, 3.4)]

Стандартный вывод
1. 3.4, 7.3, 2.5 = 38.96 2. 6.7, 6.0, 2.6 = 28.74 3. 9.6, 0.2, 2.8 = 42.96 4. 9.5, 1.9, 2.1 = 41.88 5. 0.9, 9.2, 3.4 = 68.42

Часть 2. Генерация мира

Продолжаем помогать группе исследователей.

Работоспособность автономной системы при помощи симулятора `Gazebo` проверили. Настало время проверить программный код для автоматической генерации мира в `Gazebo` (согласно условиям задания IV.11, часть 1).

За основу необходимо взять мир `clover/clover_simulation/resources/worlds/ /clover_aruco.world` и добавить туда необходимые объекты.

Формат входных данных

Набор точек, в которых необходимо собирать данные в формате: `[(x, y, z), ...]`.

Пример:

```
[(3.4, 7.3, 2.5), (6.7, 6.0, 2.6), (9.6, 0.2, 2.8), (9.5, 1.9, 2.1), (0.9, 9.2,
↪  3.4)]
```

Формат выходных данных

На стандартный вывод (`stdout`) необходимо вывести `xml` сгенерированного мира.

Решение

Для того, чтобы решить данную задачу:

1. Необходимо установить симуляционную среду Gazebo, разобраться с тем, как создается в нем мир, чтобы разработать программный код для автоматического создания мира.
2. Подготовить программный код для выполнения квадрокоптером миссии в автономном режиме согласно условиям задачи.

Пример программы-решения

Ниже приведен программный код для автономной генерации мира в симуляторе Gazebo.

```
COORDS = [(5.7, 0.7, 2.2), (6.5, 8.2, 1.1), (1.8, 7.6, 2.4), (0.2, 3.6, 2.4),
↪  (3.7, 5.8, 1.3)]

TEMPLATE = '''
   <model name='box{i}'>
     <pose>{x} {y} {z} 0 0 0</pose>
     <link name='link'>
       <collision name='collision'>
         <geometry>
           <box>
             <size>{xx} {yy} {zz}</size>
           </box>
         </geometry>
       </collision>
       <visual name='visual'>
         <geometry>
           <box>
             <size>{xx} {yy} {zz}</size>
           </box>
         </geometry>
         <material>
           <script>
             <name>Gazebo/Grey</name>
             <uri>file://media/materials/scripts/gazebo.material</uri>
           </script>
         </material>
       </visual>
       <static>1</static>
     </link>
   </model>
'''

for i, e in enumerate(COORDS):
   print(TEMPLATE.format(i=i, x=e[0], y=e[1], z=0, xx=1, yy=1, zz=e[2] - 0.2))
```

Ниже приведен программный код на языке Python 3 для выполнения квадрокоптером миссии в автономном режиме.

```
import rospy
from clover import srv
from std_srvs.srv import Trigger
from std_msgs.msg import Float64
from mavros_msgs.srv import CommandLong
import math

def navigate_wait(x=0, y=0, z=0, yaw=float('nan'), speed=2,
↪  frame_id='aruco_map_detected', auto_arm=False, tolerance=0.2):
```

```
    print(navigate(x=x, y=y, z=z, yaw=yaw, speed=speed, frame_id=frame_id,
    ↪  auto_arm=auto_arm))

    while not rospy.is_shutdown():
        telem = get_telemetry(frame_id='navigate_target')
        if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) < tolerance:
            break
        rospy.sleep(0.2)

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)
arming = rospy.ServiceProxy('/mavros/cmd/command', CommandLong)

COORDS = [(5.7, 0.7, 2.2), (6.5, 8.2, 1.1), (1.8, 7.6, 2.4), (0.2, 3.6, 2.4),
↪  (3.7, 5.8, 1.3)]
sens = []

def report():
    for i in range(len(sens)):
        print(f'{i+1}. {COORDS[i]} = {sens[i]}')

print('takeoff')
navigate_wait(z=1.0, frame_id='body', speed=3.0, auto_arm=True)
for e in COORDS:
    print('fly to point', e)
    navigate_wait(x=e[0], y=e[1], z=e[2] + 0.5)
    rospy.sleep(7)
    print('land')
    land()
    rospy.sleep(3)
    print('disarm')
    arming(command=400, param2=21196)
    rospy.sleep(5)
    sen = rospy.wait_for_message('/sensor', Float64)
    sens.append(sen.data)
    print('sen', sen.data)
    report()
    rospy.sleep(5)
    print('takeoff')
    navigate(z=1.0, frame_id='body', yaw=float('nan'), speed=3.0, auto_arm=True)
    rospy.sleep(5)

navigate_wait(x=0, y=0, z=1)
land()
report()
```