# Командный практический тур

## *Задача III.2.1. ( баллов)*

### *Требования к команде и к участникам*

Рекомендуем состав команды из 4 человек.

- Электронщик. Навыки пайки, разработки электрических схем, разработки документации.
- Конструктор. Навыки работы с инструментом, проектирования в САПР, разработки документации.
- Программист микроконтроллеров/микрокомпьютеров. Язык Python. Принципы работы микроконтроллеров/микрокомпьютеров.
- Программист в MUR IDE. Язык Python. Алгоритмы автономного управления, компьютерного зрения.

### *Оборудование и программное обеспечение*

Для изготовления и монтажа системы беспроводной зарядки можно использовать комплектующие, которые предоставили организаторы. Иные комплектующие можно использовать по согласованию с руководителем профиля. Список предоставляемых комплектующих:

| № | Наименование | Кол-во |
|---|---|---|
| Комплектующие | | |
| 1. | АНПА с двумя движителями | 12 |
| 2. | Разъемы 5pin | 12 |
| 3. | Кабель силиконовый | 12 |
| 4. | Катушка индуктивности «передатчик» | 12 |
| 5. | Катушка индуктивности «приемник» | 12 |
| 6. | Разъем-включения 2pin | 12 |
| 7. | Светодиод | 24 |
| 8. | Конденсаторы | Уп. |
| 9. | Резисторы 1кОм, 10кОм, 20кОМ, 200Ом | Уп. |
| 10. | Диод | Уп. |
| 11. | Зарядное устройство 12В | 11 |
| 12. | Листовой вспененный ПВХ 30х50 | 10 |
| 13. | Герметик двухкомпонентный | Уп. |
| 14. | Припой | 5 |
| 15. | Канифоль | 5 |
| 16. | Стержень клеевой | 25 |
| 17. | Перчатки медицинские | Уп. |
| 18. | Изолента | 7 |
| 19. | Стяжки пластиковые | Уп. |
| 20. | Магниты неодимовые | 48 |
| Оборудование | | |
| 1. | Термоклеевой пистолет | 5 |
| 2. | Паяльная станция | 5 |
| 3. | Мультиметр | 5 |
| 4. | Источник питания лабораторный | 3 |
| 5. | Кусачки | 5 |
| 6. | Плоскогубцы малые | 5 |
| 7. | Весы | 1 |
| 8. | Стриппер | 5 |
| 9. | Нож строительный | 8 |

Для программирования робота используется симулятор MUR IDE (последняя версия, доступна по ссылке MUR IDE). Разрешено использовать другие среды программирования. Команды используют собственные ноутбуки.

### *Описание задачи заключительного этапа*

По легенде в заливе у донной обсерватории перебило кабель электропитания. И она не может отправить важные данные на сервер. Но у обсерватории имеется порт беспроводной зарядки. Ваша задача — разработать передатчик беспроводной зарядки и установить его на АНПА. Также необходимо предварительно протестировать свою зарядку, сделав и приемник. После этого с помощью АНПА найти донную обсерваторию по специальным меткам и зарядить ее, чтобы она смогла отправить данные на сервер.

Задача заключительного этапа олимпиады состоит из следующих подзадач:

- Разработка и интеграция беспроводной зарядки в АНПА.

- Разработка и изготовление беспроводного приемника.
- Выполнение заданий в бассейне.
- Разработка структурной электрической схемы приемника и передатчика.
- Разработка чертежа общего вида АНПА с установленным передатчиком.

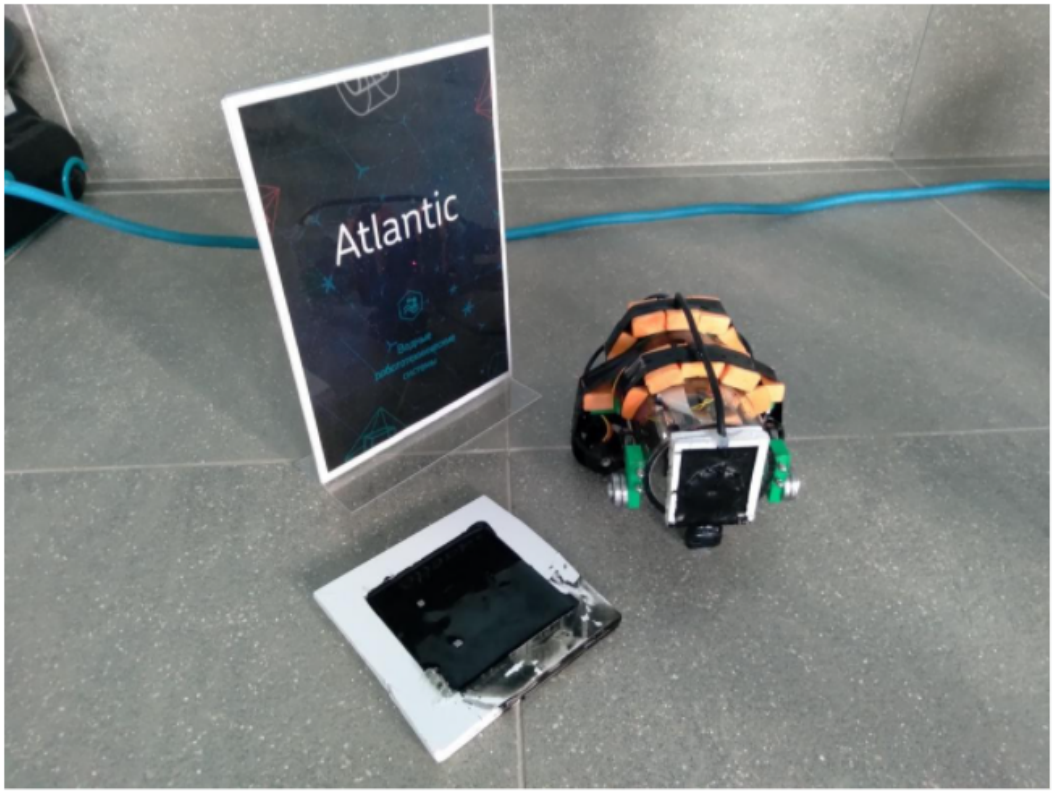
### *Порядок выполнения командной задачи*

1. В первый день соревнований каждая команда получит АНПА и комплектующие для изготовления беспроводной зарядки и приемника.
2. Команда должна разработать беспроводные приемник и передатчик и интегрировать передатчик в АНПА. В первый и второй соревновательный день команды будут размещены в мастерских. Получить баллы за данные подзадачи можно в любое соревновательное время до финальных заплывов. У команды есть две попытки сдать эти подзадачи.
3. Команды должны запрограммировать робота на выполнение подводных задач в бассейне. Команды могут разделиться: часть команды останется в мастерских, а часть будет тренироваться в бассейне. Бассейн будет доступен на 3 и 4 соревновательный день. Тренировки проходят в порядке живой очереди.
4. Подводные задачи можно сдать в последний соревновательный день во время финальных заплывов. Каждой команде будет дано 2 попытки на выполнение подводных задач.
5. Структурную электрическую схему и чертеж общего вида можно сдать в любое время до начала финальных заплывов. Данные подзадачи нельзя пересдавать.
6. Финальные заплывы начнутся в последний соревновательный день.

### *Правила, права и обязанности участников заключительного этапа олимпиады*

1. Всем командам выдается одинаковые АНПА и набор комплектующих. Основные комплектующие, такие как микрокомпьютер, батарея, двигатель, датчики, приемопередатчики не заменяются. Если у команд выйдут из строя эти комплектующие, то им придется справляться без них. Дополнительные комплектующие, такие как резисторы, светодиоды, провода можно будет в любой момент заменить.
2. Участники могут использовать собственное оборудование и/или инструменты, однако, тогда должны делиться им с другими участниками.
3. Участники не могут использовать собственные комплектующие и материалы. Только те, которые предоставили или разрешили использовать организаторы.
4. Документация сдается в электронном виде.
5. В бассейне необходимо находиться только в сменной форме, а именно в сланцах (не в кроссовках, кедах или другой обуви). Рекомендуем взять легкую одежду в бассейне (футболка, шорты).
6. Чтобы научиться программировать бортовой микрокомпьютер на функции, которые не охватывает MUR IDE рекомендуем ознакомиться с Приложением 3.

Пример разработки и изготовления зарядки и приемника.


Atlantic


Enigma

# Приложение 1. Критерии оценки структурной электрической схемы и пример решения

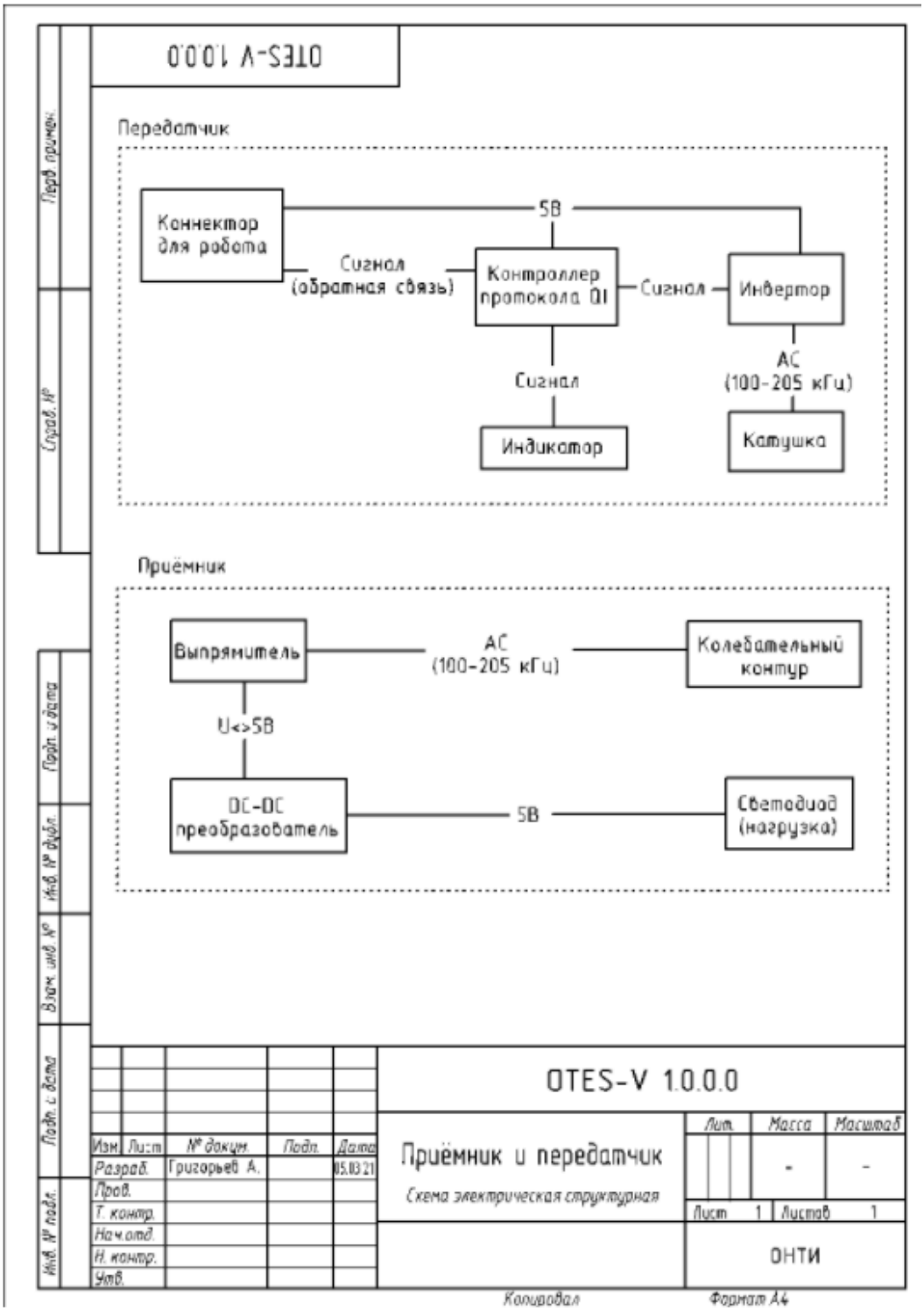
Данная схема разрабатывается в соответствии с ГОСТ 2.702-2011. Однако в целях обучения мы немного упростили требования к ней. И свели их к следующим критериям.

Критерии оценки:

- Наличие рамки для заглавного листа в соответствии с ГОСТ 2.104-2006 — 1 балл.
- Рамка заполнена в соответствии с ГОСТ 2.104-2006 — 1 балл.
- Схема выполнена в САПР — 1 балл.
- Линии связи должны состоять из горизонтальных и вертикальных отрезков и не имеют изломов и взаимных пересечений — 1 балл.
- Все линии имеют подписи и все текстовые данные, относящиеся к линиям, ориентируют параллельно горизонтальным участкам соответствующих линий — 1 балл.
- Все надписи выполнены шрифтом ГОСТ тип Б — 1 балл.
- Схема выполнена на одном листе А4 — 1 балл.
- Схема занимает более 30% площади листа — 1 балл.
- Все блоки схемы имеют названия, отражающие функциональное обозначение компонентов и их количество не избыточно 1-5 балла.
- Схема не имеет избыточных соединений между блоками 1-2 балла.

Итого можно заработать 15 баллов за схему, полностью соответствующую нашим критериям.

Пример решения (Команда OTES-V):


OTES-V 1.0.0.0
Передатчик
Коннектор для работа
5В
Сигнал (обратная связь)
Контроллер протокола QI
Сигнал
Инвертор
Сигнал
AC (100-205 кГц)
Индикатор
Катушка
Приёмник
Выпрямитель
AC (100-205 кГц)
Колебательный контур
U<>5В
DC-DC преобразователь
5В
Светодиод (нагрузка)
OTES-V 1.0.0.0
Приёмник и передатчик
Схема электрическая структурная
Лит.
Масса
Масштаб
Лист 1
Листов 1
ОНТИ
Изм. Лист № докум. Подп. Дата
Разраб. Григорьев А. 05.03.21
Пров.
Т. контр.
Нач.отд.
Н. контр.
Утв.
Копировал
Формат А4

# Приложение 2. Критерии оценки чертежа общего вида и пример решения

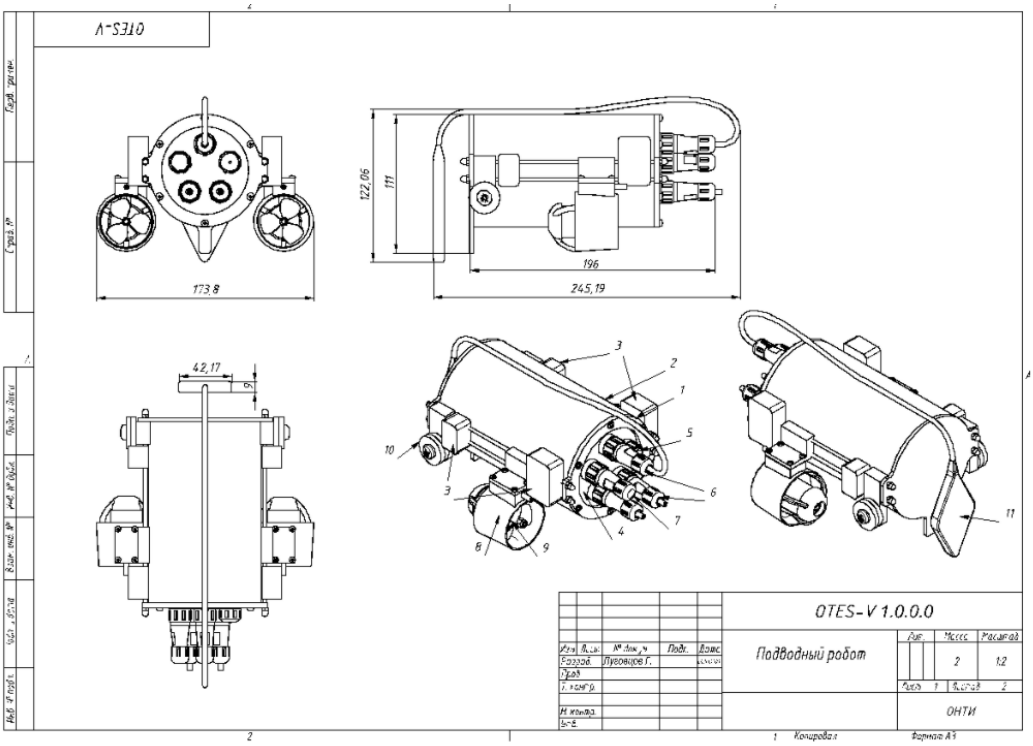
Данный чертеж разрабатывается в соответствии с ГОСТ 2.119-73 и ГОСТ 2.120-73. Однако в целях обучения мы немного упростили требования к нему. И свели их к следующим критериям.

Критерии оценки:

- Наличие рамки для заглавного листа в соответствии — 1 балл.
- Чертеж выполнен в САПР, представлен в формате PDF — 1 балл.
- Соблюден масштаб — 1 балл.
- Все основные детали обозначены — 2 балл.
- Чертеж выполнен на одном листе А4 или А3 — 1 балл.
- Представлены 3 вида и изометрия — 4 балла.
- Чертеж дополнен правильно оформленной спецификацией (все детали, кроме зарядки, должны быть в категории покупные детали) с корректно указанным количеством — 2 балла.
- Качество исполнения чертежа — 3 балла.

Итого можно заработать 15 баллов за чертеж, полностью соответствующий нашим критериям.

Пример решения (Команда OTES-V):



| Форм. | Зона | Поз. | Обозначение | Наименование | Кол. | Примеч. |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | Покупные изделия | | |
| | | | | | | |
| | | 1 | OTES-V 1.1.1.0 | Заглушка | 1 | |
| | | 2 | OTES-V 1.1.2.0 | Акриловый корпус | 1 | |
| | | 3 | OTES-V 1.1.3.0 | Элемент плавучести | 4 | |
| | | 4 | OTES-V 1.1.4.0 | Крышка с гермовводами | 1 | |
| | | 5 | OTES-V 1.2.1.0 | Заглушка | 1 | |
| | | 6 | OTES-V 1.2.2.0 | Гермоввод | 3 | |
| | | 7 | OTES-V 1.2.3.0 | Перемычка | 1 | |
| | | 8 | OTES-V 1.3.1.0 | Корпус-насадка | 2 | |
| | | 9 | OTES-V 1.3.2.0 | Движитель | 2 | |
| | | 10 | OTES-V 1.4.1.0 | Балласт | 4 | |
| | | | | Разрабатываемые изделия | | |
| | | 11 | OTES-V 1.5.1.0 | Передатчик энергии | | |
| | | | | беспроводной | 1 | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Изм | Лист | № докум. | Подп. | Дата | OTES-V 1.0.0.0 | Лист 2 |
|---|---|---|---|---|---|---|

# Приложение 3.

Для использования GPIO-портов на аппарате имеется предустановленная Python-библиотека pigpio. Данная библиотека позволяет работать с GPIO на достаточно низком уровне, включая ввод и вывод сигнала, использование ШИМ и прерываний.

В качестве примера работы с библиотекой напишем простой скрипт для управления GPIO-портом 22, а именно включение и выключение с задержкой в секунду:

```
import pigpio
import time
gpio = pigpio.pi()

while True:
    gpio.write(22, 1)
    time.sleep(1)

    gpio.write(22, 0)
    time.sleep(1)
```

Рекомендуется ознакомиться с документацией по библиотеке pigpio (http://abyz.me.uk/rpi/pigpio/python.html).

Также рекомендуется изучить документацию по библиотеке машинного зрения OpenCV (https://docs.opencv.org/master/d6/d00/tutorial_py_root.html) и серию видеоуроков по программированию в MUR IDE (https://www.youtube.com/playlist?list=PLY0BZJ_WsvZLeZN-JVljgi5-Vvyj10Cd0). Для тестирования алгоритмов машинного зрения в симуляторе можно создать собственные сцены для симулятора MUR IDE, используя материалы видеоурока (https://www.youtube.com/watch?v=K8Wc4A1U5XM&list=PLY0BZJ_WsvZLeZN-JVljgi5-Vvyj10Cd0&index=9).

Пример программного кода на выполнение задания в бассейне (Команда Perpetuum Mobile Squad):

```
1  import pymurapi as mur
2
3  SIM = False
4  if not SIM:
5      import pigpio
6  import time
7  import cv2 as cv
8  import typing
9  from math import hypot
10 import numpy as np
11
12 # HSV
13 green_lower_hsv = (72 // 2, 100, 30 * 255 // 100)
14 green_upper_hsv = (175 // 2, 255, 60 * 255 // 100)
15
16 # LAB
17 green_lower_lab = (0, 85, 130)
18 green_upper_lab = (255, 105, 190)
19
20 auv = mur.mur_init()
21 if not SIM:
22     gpio = pigpio.pi()
23     gpio.write(22, 1)
```

```
24      vid = cv.VideoCapture(0)
25
26
27  def set_motors(left, right):
28      left = max(min(left, 1), -1)
29      right = max(min(right, 1), -1)
30      print('setting', left, right)
31      if not SIM:
32          auv.set_motor_power(0, right * 43)
33          auv.set_motor_power(1, -left * 50)
34      else:
35          auv.set_motor_power(0, left * 30)
36          auv.set_motor_power(1, right * 30)
37
38
39  def go(power_fraction, diff=0.):
40      set_motors(power_fraction - max(diff, 0), power_fraction - max(-diff, 0))
41
42
43  def stop():
44      set_motors(0, 0)
45
46
47  def show():
48      if SIM:
49          cv.imshow('Green', get_image())
50          cv.waitKey(1)
51
52
53  def total_green(img) -> int:
54      return np.sum(get_image())
55
56
57  def get_image():
58      curr_img = vid.read()[1]
59
60      img_orig_hsv = cv.cvtColor(curr_img, cv.COLOR_BGR2HSV)
61      img_orig_lab = cv.cvtColor(curr_img, cv.COLOR_BGR2LAB)
62      img_orig_rgb = cv.cvtColor(curr_img, cv.COLOR_BGR2RGB)
63
64      r = img_orig_rgb[:, :, 0]
65      g = img_orig_rgb[:, :, 1]
66      b = img_orig_rgb[:, :, 2]
67
68      img_rgb = np.uint8((g > 1.1 * r) & (g > 1.05 * b) & (g > (r + b) * 0.9)) * 255
69
70      img_hsv = cv.inRange(img_orig_hsv, green_lower_hsv, green_upper_hsv)
71      img_lab = cv.inRange(img_orig_lab, green_lower_lab, green_upper_lab)
72
73      return img_hsv & img_lab & img_rgb
74
75
76  def go_to_mass_center(img) -> float:
77      _, contours, _hierarchy = cv.findContours(get_image(), cv.RETR_TREE,
        ↪  cv.CHAIN_APPROX_SIMPLE)
78      arrow_candidates = list(get_arrow_candidates(contours))
79      if not len(arrow_candidates):
80          return 0
81      arrow = max(arrow_candidates, key=lambda cand: cand[1]['m00'])[0]
82      moments = cv.moments(arrow)
```

```
83      area = moments['m00']
84      cx, cy = int(moments['m10'] / area), int(moments['m01'] / area)
85      cx -= len(img[0]) // 2
86      cy -= len(img) // 2
87      go(-cy * 0.005, -cx * 0.004)
88      print(cx, cy)
89      return hypot(cx, cy)
90
91
92  def get_distance_from_line(point_1, point_2, center):
93      return ((point_2[1] - center[1]) / (point_1[1] - center[1]) * (point_1[0] -
        ↪  center[0]) - point_2[0] + center[0]) / (
94              (point_2[1] - center[1]) / (point_1[1] - center[1]) - 1)
95
96
97  def get_arrow_from_cnt(contour, img_shape):
98      rect = cv.minAreaRect(contour)
99      points = cv.boxPoints(rect)
100     center = (img_shape[0] / 2, img_shape[1] / 2)
101     long_lines_second = hypot(points[2][0] - points[1][0], points[2][1] -
        ↪  points[1][1]) > hypot(
102         points[1][0] - points[0][0], points[1][1] - points[0][1])
103     points = points if not long_lines_second else (list(points[1:]) + [points[0]])
104     dist = (get_distance_from_line(points[1], points[0], center)
105     + get_distance_from_line(points[3], points[2], center)) / 2
106     angle = (rect[2] + 90 * long_lines_second) * 2 * 3.1415 / 180
107     # print('Raw angle:', rect[2], 'w, h:', rect[1])
108     return angle, dist if 'nan' not in str(dist).lower() else 0
109
110
111 def get_arrow_candidates(contours):
112     for cnt in contours:
113         moments = cv.moments(cnt)
114         area = moments['m00']
115         # perimeter = cv.arcLength(cnt, True)
116         _, (w, h), _ = cv.minAreaRect(cnt)
117         if area >= 20 and w * h / area <= 1.4 and max(w, h) / min(w, h) >= 1.3:
118             # print('we have a candidate')
119             yield cnt, moments
120
121
122 def is_charging(recheck_time=None) -> bool:
123     if SIM:
124         return False
125     result = bool(round(gpio.read(4)))
126     if recheck_time is None or not result:
127         return result
128     time.sleep(recheck_time)
129     if bool(round(gpio.read(4))):
130         return True
131
132
133 def locate_arrow(img) -> typing.Optional[typing.Tuple[float, int]]:
134     # -> (angle, distance from the center of the screen to the center of intersection
        ↪  of the arrow with the horizontal
135     # central line)
136     _, contours, _hierarchy = cv.findContours(img, cv.RETR_TREE,
        ↪  cv.CHAIN_APPROX_SIMPLE)
137     arrow_candidates = list(get_arrow_candidates(contours))
138     if not len(arrow_candidates):
```

```
139         return None
140     # print('len(arrow_candidates):', len(arrow_candidates))
141     arrow, area = max(arrow_candidates, key=lambda cand: cand[1]['m00'])
142     # print('Arrow area:', area['m00'])
143     # moments = cv.moments(arrow)
144     # area = moments['m00']
145     # cx, cy = int(moments['m10'] / area), int(moments['m01'] / area)
146     # (x, y), _radius = cv.minEnclosingCircle(arrow_cnt)
147     # angle = atan((cy - y) / (cx - x))
148     # delta = (sum(x * pix for x, pix in enumerate(img[len(img) // 2])) //
        ↪  len(img[0])) - len(img[0]) // 2
149     # return angle, delta
150     return get_arrow_from_cnt(arrow, img.shape)
151
152
153 def follow_arrow(angle, delta):
154     go(1, angle * 0.05 + delta * 0.05)  # watch for coefficients
155
156
157 def rotate_for_arrow(convergence_time=3.5, tol=0.06):
158     convergence_start = None
159     while True:
160         show()
161
162         res = locate_arrow(get_image())
163         if res is None:
164             time.sleep(0.1)
165             continue
166         angle, _ = res
167         if abs(angle) < tol and not convergence_start:
168             convergence_start = time.time()
169         if abs(angle) > tol:
170             convergence_start = None
171         if abs(angle) < tol and time.time() - convergence_start > convergence_time:
172             return
173         print('Angle:', angle)
174         set_motors(angle * 0.5, -angle * 0.5)
175
176
177 def go_along_arrow(timeout=2.5):
178     found_arrow = False
179     start_time = None
180     while True:
181         img = get_image()
182         arrow_data = locate_arrow(img)
183         if start_time is not None and time.time() - start_time > timeout:
184             return
185         if arrow_data is None:
186             print('We dont see the arrow')
187             go(0.5)
188             time.sleep(0.2)
189             continue
190         else:
191             print(arrow_data)
192             if not found_arrow:
193                 found_arrow = True
194                 start_time = time.time()
195         angle, delta = arrow_data
196         if not found_arrow or abs(angle) > 0.18:
197             rotate_for_arrow(convergence_time=0.4)
```

```
            print('Rotated')
            continue
        follow_arrow(angle, delta)


def go_to_charging():
    start_time = time.time()
    while not is_charging(recheck_time=0.08):
        go(0.5)
        if time.time() - start_time > 10:
            go(-1)
            time.sleep(3)
            go_to_charging()


def charge():
    go(0.15)
    time.sleep(15)
    go(-1)


if __name__ == '__main__':
    print('a')
    rotate_for_arrow()
    print('Going to the center of masses')
    while go_to_mass_center(get_image()) > 20:
        show()
    stop()
    print('Done')
    print('Green mass:', total_green(get_image()))
    # old_green = total_green(get_image())
    rotate_for_arrow(convergence_time=5)
    print('Rotated again')
    # go_along_arrow()
    # print('The arrow has ended. Green mass:', total_green(get_image()))
    go(0.8)
    time.sleep(3.5)
    while total_green(get_image()) < 800000:
        show()
        go(0.5)
    print()
    # rotate_for_arrow()
    while go_to_mass_center(get_image()) > 20:
        show()
    rotate_for_arrow()
    while go_to_mass_center(get_image()) > 20:
        show()
    rotate_for_arrow()
    # go_along_arrow()
    print('The second arrow has ended')
    go(0.7)
    time.sleep(3.5)
    while total_green(get_image()) < 800000:
        show()
        go(0.5)
    rotate_for_arrow()
    while go_to_mass_center(get_image()) > 20:
        show()
    rotate_for_arrow()
    go_to_charging()
```

```
258         charge()
```