Передовые производственные технологии

2022/23 учебный год

Инженерный тур

Общая информация

Участникам финального этапа предлагается применить передовые производственные технологии при реализации поисково-спасательного робота для работы в замкнутых помещениях.

Легенда задачи

Развитие технологий всегда давало новые удобства и новые возможности человеку и его среде обитания. Электрификация, газификация, теплые дома, бережливые производства. Однако, у всего есть и обратная сторона — повышенная опасность. К сожалению, происшествия, связанные с утечками опасных газов (не только бытового, но и многих других) стали происходить все чаще и чаще и если обследовать, например, открытую и доступную трубу в квартире или на заводе довольно просто, то, при обследованиях и обнаружениях утечек в труднодоступных местах практически всегда возникают риски и проблемы.

На помощь человеку и борцам с происшествиями давно приходят передовые технологии, в том числе и производственные — современное проектирование, симуляция, автоматизация, новые материалы и технологии производства, которые помогают не только уменьшить возможные последствия аварий, но и полностью их избежать.

В решении проблемы поиска и локализации утечки опасных газов могут помочь современные малогабаритные роботы, которые могут проворно пробраться в небольшие пространства, где не поместится физически ни один взрослый или ребенок.

И наша задача — создать (доработать) такого робота!

Требования к команде и компетенциям участников

Количество участников в команде: 3.

Роли:

- **Схемотехник**. В задачу данного участника команды входит решение задач, связанных с построением электрических схем и программированию контроллеров типа Arduino. От него необходимо понимание того, каким образом строятся электрические схемы, как и к каким портам подключать различные устройства при работе контроллера, а также уметь программировать на языке Arduino контроллеры данного семейства.
- **Конструктор**. В задачу данного участника входит создание 3-х мерных конструкций с применением средств автоматизированного проектирования. От него необходимо знания средств автоматизированного проектирования (САПР, CAD) типа Fusion 360, понимание особенностей проектирования моделей для 3D печати.

- **Программист-алгоритмист**. В задачу данного участника входит решение задач на языке высокого уровня с использованием языка Python на микрокомпьютере под управлением ос Lunux типа Ubuntu. От него необходимо знание языка Python, умение строить алгоритмы для написания программ, умение отлаживать их и тестировать.
- **Менеджер/капитан команды** (роль, выполняемая по совместительству одним из участников команды). В задачу данного участника входит руководство разработкой в целом, распределение ролей в команде, а также составление демонстрационных материалов, проведение демонстраций и защита проекта.

Участники вольны менять роли внутри команды, а также помогать друг другу внутри команды по смежным ролям.

Оборудование и программное обеспечение

Общее:

1. Сверлильный станок — 1 шт.
2. 3D принтеры — мин. 5 шт.
3. PLA-пластик 1,75 мм (1 кг) зеленый — 3 шт.
4. PLA-пластик 1,75 мм (1 кг) желтый — 3 шт.
5. BBJ-40 MF, 30 см; шлейф с разъемами папа-мама 40 шт — 15 шт.
6. BBJ-40 FF 30 см; шлейф с разъемами мама-мама 40 шт — 5 шт.
7. BBJ-40 MM 30 см; шлейф с разъемами папа-папа 40 шт — 5 шт.
8. Набор сверл по металлу P6M5 из 13 шт — 3 шт.
9. Щетка-сметка и совок — 2 шт.

Командное (выдается каждой команде):

1. Микроконтроллер Arduino MEGA 2560 — 1 шт.
2. Монитор — 1 шт.
3. Клавиатура — 1 шт.
4. Мышь — 1 шт.
5. DFR0315, RPLIDAR LASER SCANNER DEV KIT — 1 шт.
6. VL6180X лазерный датчик расстояния — 8 шт.
7. Мотор с редуктором JGA25-370 12 В соотношение 1 : 78 108 об/мин — 2 шт.
8. Motor Control Board, Плата управления DC и шаговыми двигателями — 1 шт.
9. Блок питания, 12 В,2 А,24 Вт (адаптер), штекер $5,5 \times 2,1/10$ — 1 шт.
10. BW1411, Кабель мультимедийный USB2.0 A вилка — USB B вилка, 1,8 м — 1 шт.
11. Troyka-Encoder, Энкодер для Arduino проектов — 2 шт.
12. KLS5-18650-2W-L150 (TBH-18650-2B-W) (FC1-5217), Батарейный отсек 2×18650 — 1 шт.
13. 18650, Аккумулятор Li-ion 18650, 2800mAh, 3,7 V, с защитой — 2 шт.
14. Orange Pi 3–1 шт.
15. Адаптер/блок питания с USB разъемом, 5 В,3 А,15 Вт — 1 шт.
16. PLS-40 (DS1021-1×40), Вилка штыревая 2,54 мм 1×40pin прямая — 3 шт.

17. PLS-40R (DS1022-1×40R), Вилка штыревая 2.54мм 1×40pin угловая — 1 шт.
18. Флюс ЛТИ-120 LUX — 1 шт.
19. Припой без канифоли ПОС-61, 10 г, ø1.0 мм — 1 шт.
20. Винт DIN 7985 полусфера, шлиц Ph, цинк M3×8 уп. — 35 шт — 2 шт.
21. Винт DIN 7985 полусфера, шлиц Ph, цинк M3×16 уп. — 25 шт — 2 шт.
22. Винт DIN 7985 полусфера, шлиц Ph, цинк M3×25 уп. пакет малый — 18 шт — 2 шт.
23. Винт DIN 965 потай, шлиц Ph, цинк M3×10 уп. пакет малый — 30 шт — 1 шт.
24. Винт DIN 965 потай, шлиц Ph, цинк M3×12 уп. — 25 шт — 1 шт.
25. Винт DIN 965 потай, шлиц Ph, цинк M3×20 уп. пакет малый — 20 шт 1шт.
26. Гайка DIN 934, цинк M 3 уп. — 50 шт — 4 шт.
27. Шайба DIN 125 простая, цинк M 3 уп. — 60 шт — 1 шт.
28. Очки защитные открытого типа, прозрачные, ударопрочный поликарбонат — 2 шт.
29. Перчатки трикотажные, рисунок ПВХ — 3 шт.
30. ZD-6 (MAX 40W), Пистолет клеевой 40 Вт — 1 шт.
31. Стержни клеевые ø 11 мм, 270 мм, прозрачные (хедер) (10 шт/уп) — 1 шт.
32. Клей моментальный СЕКУНДА3, 3 г — 2 шт.
33. Клей Момент Кристалл прозрачный 30мл — 1 шт.
34. Выключатель-кнопка металл 220 V 2 A (2 с) OFF-(ON) ø7,2 красная Micro (RWD-301, PBS-10B) — 2 шт.
35. Выключатель-кнопка 250 V 1 A (2 с) ON-OFF желтая (PBS-11A) — 2 шт.
36. KS-6346, Коврик для пайки термостойкий силиконовый — 1 шт.
37. Нож технический 18 мм усиленный прорезиненный, кассета 3 лезвия — 1 шт.
38. Изолента ПВХ 15 мм × 20 м синяя — 1 шт.
39. Лента малярная, 30×15, на бумажной основе — 1 шт.
40. Коврик (мат) для резки 3-слойный, А3 (450×300 мм), настольный, зеленый, 3 мм, KW-trio — 1 шт.
41. Оплетка для удаления припоя, медная, 2,5 мм × 1,5 м — 1шт.
42. Набор хомутов цветных пластиковых НХ-2 — 1 шт.
43. Паяльник ОНЛАЙТ 80 835 OSE-Pes02-40W-IP — 1 шт.
44. Набор водостойкой наждачной бумаги HANDSIZE FULL SET-1 140×115 мм AF-HS-SET-1 — 2 шт.
45. «Электрик», Набор термоусадочной трубки в тубе — 1 шт.
46. Зажим винтовой 0,75–4 мм^2 12 пар — 1 шт.
47. ПГВА-10-0.5, Провод акустический ШВПМ 5 м (сечение 2×0,75 мм кв.) АЭНК — 1 шт.
48. Скотч двухсторонний 20 мм×5 м белый ABRO — 1 шт.
49. Набор отверток 4 шт.: SL5×75, SL6×100, PH1×75, PH2× 1 шт.
50. Набор надфилей, 160 × 4 мм, 6 шт., обрезиненные рукоятки — 1 шт.
51. Шлейф цветной RCA 16 × 0,08 мм 1 метр — 2 шт.
52. Преобразователь напряжения для питания Orange PI

Программное обеспечение:

1. САПР Autodesk Fusion360.
2. Arduino IDE.
3. Ubuntu 20.04.
4. ROS Noetic.
5. Visual Studio Code.
6. Pycharm.

Описание задачи

Командам необходимо «довести» платформу колесного робота для решения задачи поиска утечки опасных газов в лабиринте. Участникам дана платформа колесного робота, с предварительно установленными на нее (но не соединенными) элементами системы управления нижнего уровня (микроконтроллер, двигатели, колеса, драйверы двигателей), и набор датчиков и устройств для организации верхнего уровня и системы датчиков окружающего пространства. Также на нижнем (Arduino) и верхнем (Orange PI) программном уровне дана заготовка ПО, в которой отсутствует модуль автоматической навигации в лабиринте.

От участников потребуется разработать систему управления роботом для прохода лабиринта, компоновку верхней части платформы для установки предложенных датчиков и устройств, а также схему соединения и элементы соединения датчиков и устройств.

Лабиринт будет доступен участникам для осмотра и тестирования на протяжении всего периода разработки, но с ограниченным количеством попыток запуска в первые 2 дня. У лабиринта будет 1 вход и 1 выход и точки для «впрыскивания» «опасного газа».

Робот должен проследовать в лабиринте от входа, зафиксировать время контакта с «утечкой» «опасного газа» и выехать на выходе.

Участники вольны в выборе:

- Количества и расположения датчиков и их соединения (схемотехник/аппаратчик).
- Алгоритме прохождения маршрута (программист).
- Конструкции верхней части робота, удерживающей все датчики и устройства (конструктор).

Общая деятельность в рамках инженерного тура разбита на 3 основных этапа, в рамках этапов существуют 2 дедлайна — 1 мягкий и 1 жесткий дедлайн.

Этап 1

На первом этапе участникам предложено составить мини-презентацию разработанной концепции, в которой необходимо будет указать и обосновать свой выбор конструкции, расположение датчиков, их количество и общую реализуемость конструкционных элементов для дальнейшей 3D печати, а также обосновать выбор алгоритма движения внутри лабиринта.

Принимая во внимание, что в ходе непосредственной разработки, концепция может претерпеть изменения, дальнейшая доработка концепции после дедлайна не штрафуется.

Этап 2

Участники должны представить реализованное решение, удовлетворяющее следующим требованиям:

Проработать совместно компоновку датчиков и устройств для решения задачи ориентации в замкнутом лабиринте, удовлетворяющее требованиям:

- должно быть произведено методом аддитивного производства;
- должно надежно крепить все выбранные устройства и датчики;
- должно обеспечивать прокладку кабелей и шин связи;
- количество и состав датчиков должны позволять выполнить задачу ориентации в пространстве (лабиринте), кроме того, должна быть реализована двухуровневая система управления;
- соединение устройств не должно приводить к уничтожению оборудования.

С помощью современной САПР (Fusion360) разработать конструкцию верхней крышки, содержащую крепления для датчиков (дистанции, опасных газов, лидара) и устройств (Orange PI) и подготовить к 3D печати, удовлетворяющие требованиям:

- используемый тип пластика PLA или ABS, метод печати — FDM;
- минимизация использования поддержек во избежание проблем с печатью;
- габариты не более 180×180 мм;
- количество деталей — не более 5.

Создать и реализовать схему соединения для датчиков и устройств. Припаять и соединить все, что необходимо, провести провода, правильно подключить устройства и датчики. Запрограммировать контроллер нижнего уровня, которым является Arduino на прием и отправку данных с датчиков и работу с двигателями постоянного тока, удовлетворяющую требованиям:

- должен быть обеспечен обмен данными между датчиками и устройствами робота;
- должен быть обеспечен обмен данными с верхним уровнем;
- все заложенные устройства должны работать корректно;
- должно быть обеспечено управление двигателями постоянного тока для обеспечения возможности поворота и движения во всех направлениях.

Реализовать систему автоматической ориентации в замкнутом пространстве, которая:

- должна обеспечивать передвижение робота в замкнутом пространстве без контакта с объектами лабиринта;
- должна обеспечивать выход из лабиринта не в точке входа;
- должна обеспечивать обмен данными с нижним уровнем;
- должна обеспечить фиксацию времени обнаружения опасных газов.

Не допускается доработка готового устройства после наступления дедлайна по этапу 2.

Итоговые испытания

Для комплексной оценки решения задачи, участникам предстоит запустить робота в лабиринт размером $3 \times 3,5$ м, представляющий собой последовательность узких коридоров, шириной 0,4–0,5 м с одним входом и одним выходом. Задача робота при запуске его со входа, полностью пройти лабиринт и выйти на выходе. При прохождении в лабиринте, в одной из точек будет «впрыснут» «опасный газ», и робот на выходе должен вывести время контакта с утечкой «опасного» газа в лабиринте от момента старта.

При этом оценивание решение основывается на следующих критериях:

1. Прохождение лабиринта до конца.
   Прохождение лабиринта до конца подразумевает проезд робота от входа в лабиринт до выхода из него. Чем быстрее робот выйдет из лабиринта, тем лучше. Максимальное время прохождения лабиринта 15 минут (900 с). Из этого времени вычитается фактическое время прохождения лабиринта командой (Пример: робот прошел лабиринт за 9 минут 38 секунд, это 578 секунд. $900 - 578 = 322$), далее результат делится на 100 (итог 3,22). Полученный результат прибавляется к общим баллам команды. Если роботу участников понадобилось больше времени на прохождение лабиринта, то команда не получает баллов за время. За прохождение лабиринта до конца команда получает 50 баллов.
2. Обнаружение утечки газа.
   Обнаружение утечки газа подразумевает фиксирование времени контакта с утечкой газа в разных частях лабиринта. Газ впрыскивается 1 раз. При обнаружении газа роботу необходимо обозначить время каким-либо способом. За обнаружение газов можно получить максимум 30 баллов. Если робот обнаружил газ с точностью до 2 секунд, команда не теряет баллы. Если робот обнаружил газ с точностью от 2 до 4 секунд, команда теряет 5 баллов из максимально возможных 30. В случае если время обнаружения выходит за пределы 4 секунд, команда теряет 10 баллов из максимально возможных 30. В случае отсутствия обнаружения газа команда не получает баллов вообще.
3. Избежать столкновений с элементами лабиринта.
   Робот не должен врезаться в стены лабиринта. За каждый удар о стены, у команды, собравшей робота, отнимается один балл. Столкновение засчитывается при контакте основного корпуса робота со стенами лабиринта (к примеру, контакты выступающих проводов не учитываются).
4. Остановиться при достижении финиша.
   Остановиться при достижении финиша подразумевает остановку робота после пересечения финишной линии не более чем в пятидесяти сантиметрах от нее.
   За остановку при достижении финиша в пределах пятидесяти сантиметров от финишной линии команда получает 20 баллов.

Система оценивания

Итоговый балл участника равен:

$$\text{Баллы за Физику} \times 0,2 + \text{баллы за информатику} \times 0,2 +$$

$$+\text{баллы за инженерный тур} \times 0,6.$$

Итоговый балл команды за инженерный тур: сумма баллов, набранной командой при проведении финальных испытаний.

В случае равенства баллов, участникам предлагается решить дополнительное теоретическое задание по тематике соревнования на время (не более 15 минут), которое станет решающим при выборе победителя.

По итогам определяется:

1 команда-победитель олимпиады (командный зачет).

Число призеров и победителей (суммарно) в индивидуальном зачете не более 25% от общего числа участников. Число победителей не более 8% от числа участников. Призеры и победители определяются как набравшие ряд наибольших итоговых баллов в своей возрастной группе.

Решение задачи

Ссылка на решение: https://disk.yandex.ru/d/V38iRJB4KWOM4A.

Программное обеспечение

Верхний уровень, файл автоматического управления.

Pysensors_ data_ node.py

```
#!/usr/bin/env python3
import rospy, math, time
from std_msgs.msg import String
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

#                          12
#                     13       11
#                14         ^       10
#            15           /   \         9
#         16                |              8
#     17                    |                  7
# 18                        |                      6
#     19                                       5
#         20                                 4
#             21                         3
#                 22                 2
#                     23       1
#                         24

MYROBOT_MAX_LIN_VEL = 0.22
MYROBOT_MAX_ANG_VEL = 1.0 #1.75 #2.5
SENSORS_DISTANCE = 0.2
LIDAR_DISTANCE = 1.0 #1.0 #0.4
MIN_LIDAR_DISTANCE = 0.3 #0.2
MAX_LIDAR = 0.4 #0.22
MIN_LIDAR = 0.2 #0.1
MYROBOT_SPEED = 1
SEGMENTS = 24
DIST_MAX_RANGE = 400

```

```
32  def angle(angle):
33      return MYROBOT_MAX_ANG_VEL * (angle / MYROBOT_MAX_ANG_VEL)
34
35  def logL(x, y):
36      print(f"turn left from: {x} with angle: {y}")
37
38  def logR(x, y):
39      print(f"turn right from: {x} with angle: {y}")
40
41  def lerp(a, b, coeff):
42      return float(a) + float(b-a) * float(coeff)
43
44  class KeyboardTeleop:
45      def __init__(self):
46          rospy.init_node('keyboard_teleop')
47
48          self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
49          self.sub_front = rospy.Subscriber('SensorsData', String, self.onIRSensor)
50          self.dist_front_0 = 0.0
51          self.dist_front_1 = 0.0
52          self.dist_front_2 = 0.0
53          self.gasValue = 0
54
55          self.tic_start = 0
56          self.tic_detect = 0
57          self.start_timer = False
58
59          self.sub_lidar = rospy.Subscriber('/scan', LaserScan, self.onLaserSensor)
60          self.distances = []
61          self.lidar = LaserScan.angle_max
62
63          for i in range(SEGMENTS):
64              self.distances.append(0)
65
66          self.target_linear_vel   = 0.0
67          self.target_angular_vel  = 0.0
68
69      def leftTurn(self, idx):
70          self.target_angular_vel = -self.distances[idx]
71          self.target_linear_vel = MYROBOT_MAX_LIN_VEL
72          if(self.distances[idx] >= 1.0):
73              self.target_angular_vel = 1
74
75          #self.target_angular_vel = 1.0
76          #self.target_angular_vel = (-1) * self.target_angular_vel
77          if self.target_angular_vel < -0.4:
78              self.target_angular_vel = -0.4
79          self.target_angular_vel = self.target_angular_vel / 1.5
80          logL(idx, self.target_angular_vel)
81
82      def rightTurn(self, idx):
83          self.target_angular_vel = self.distances[idx]
84          self.target_linear_vel = MYROBOT_MAX_LIN_VEL
85          #self.target_angular_vel = -1.0
86          if self.target_angular_vel > 0.4:
87              self.target_angular_vel = 0.4
88          self.target_angular_vel = self.target_angular_vel / 1.5
89          logR(idx, self.target_angular_vel)
90
91      def run(self):
```

```
92          rate = rospy.Rate(10)
93          try:
94              while not rospy.is_shutdown():
95                  self.update_velocity()
96                  self.check_gas()
97                  rate.sleep()
98          except rospy.ROSInterruptException:
99              pass
100
101     def onIRSensor(self, msg):
102         values = msg.data.split(';')
103
104         if len(values) == 7:
105            self.gasValue = float(values[6])
106            self.dist_front_0 = int(values[5]) / 1000
107            self.dist_front_1 = int(values[4]) / 1000
108            self.dist_front_2 = int(values[3]) / 1000
109         else:
110            rospy.loginfo(f"Only {len(values)} sensors are responding")
111            self.dist_front_0 = 0
112            self.dist_front_1 = 0
113            self.dist_front_2 = 0
114            self.gasValue = 0
115
116     def onLaserSensor(self, msg):
117         for i in range(SEGMENTS):
118             self.distances[i] = 9999
119
120         angle = msg.angle_min
121         for r in msg.ranges:
122             if math.isinf(r):
123                 r = 12
124
125             a = angle * 180 / math.pi
126             idx = int(a / (360 / SEGMENTS)) % SEGMENTS
127             # print(repr(idx) + ' of ' + repr(a) + ' with dist= ' + repr(r))
128             self.distances[idx] = min(self.distances[idx], r)
129             angle = angle + msg.angle_increment
130
131     def check_gas(self):
132
133         if self.gasValue >= 20:
134             if self.start_timer == False:
135                 self.start_timer = True
136                 self.tic_start = time.perf_counter()
137
138             if self.start_timer == True:
139                 self.tic_detect = time.perf_counter()
140
141         print(f"Last gas detect: {self.tic_detect - self.tic_start:0.4f} seconds")
142
143     def update_velocity(self):
144         # print('distances: ' + repr(self.distances))
145
146         #self.target_linear_vel = 0
147         #self.target_angular_vel = 0
148         print(f"Sensor left: {self.dist_front_0} ; Sensor center:
            ↪ {self.dist_front_1} ; Sensor right: {self.dist_front_2}")
149
150         # From here
```

```
151             if(self.dist_front_0 < SENSORS_DISTANCE or self.dist_front_2 <
                ↪  SENSORS_DISTANCE or self.dist_front_1 < SENSORS_DISTANCE):
152                 if self.dist_front_2 < SENSORS_DISTANCE:
153                     self.target_linear_vel = MYROBOT_MAX_LIN_VEL
154                     self.target_angular_vel = -MYROBOT_MAX_ANG_VEL * (1 -
                        ↪  self.dist_front_2)
155                 if self.dist_front_0 < SENSORS_DISTANCE:
156                     self.target_linear_vel = MYROBOT_MAX_LIN_VEL
157                     self.target_angular_vel = MYROBOT_MAX_ANG_VEL * (1 -
                        ↪  self.dist_front_0)
158                 if self.dist_front_1 < SENSORS_DISTANCE:
159                     self.target_linear_vel = MYROBOT_MAX_LIN_VEL
160                     # self.target_angular_vel = 0.0
161 
162                 print(f"From Sensor: {self.target_angular_vel}")
163             else:
164                 if((self.distances[6] > MAX_LIDAR or self.distances[5] > MAX_LIDAR or
                    ↪  self.distances[4] > MAX_LIDAR or self.distances[7] > MAX_LIDAR or
                    ↪  self.distances[8] > MAX_LIDAR) and
165                    (self.distances[6] > MIN_LIDAR and self.distances[5] > MIN_LIDAR
                       ↪  and self.distances[4] > MIN_LIDAR and self.distances[7] >
                       ↪  MIN_LIDAR and self.distances[8] > MIN_LIDAR)):
166                     if(self.distances[6] > MAX_LIDAR):
167                         self.leftTurn(6)
168                     elif(self.distances[5] > MAX_LIDAR):
169                         self.leftTurn(5)
170                     elif(self.distances[4] > MAX_LIDAR):
171                         self.leftTurn(4)
172                     elif(self.distances[7] > MAX_LIDAR + 0.1):
173                         self.leftTurn(7)
174                     elif(self.distances[8] > MAX_LIDAR + 0.2):
175                         self.leftTurn(8)
176                 elif self.distances[0] > MAX_LIDAR or self.distances[1] > MAX_LIDAR or
                    ↪  self.distances[23] > MAX_LIDAR:
177                     print('Along the wall')
178                     self.target_angular_vel = 0.0
179                     self.target_linear_vel = MYROBOT_MAX_LIN_VEL
180                 elif self.distances[21] > MAX_LIDAR or self.distances[20] > MAX_LIDAR
                    ↪  or self.distances[19] > MAX_LIDAR or self.distances[18] >
                    ↪  MAX_LIDAR or self.distances[22] > MAX_LIDAR:
181                     if(self.distances[20] > MAX_LIDAR):
182                         self.rightTurn(20)
183                     elif(self.distances[21] > MAX_LIDAR):
184                         self.rightTurn(21)
185                     elif(self.distances[22] > MAX_LIDAR):
186                         self.rightTurn(22)
187                     elif(self.distances[19] > MAX_LIDAR):
188                         self.rightTurn(19)
189                     elif(self.distances[18] > MAX_LIDAR):
190                         self.rightTurn(18)
191 
192             for i in range(SEGMENTS):
193                 print(f"Seg {i} - {self.distances[i]}")
194 
195             twist = Twist()
196             rate = rospy.Rate(10)
197             twist.linear.x = self.target_linear_vel; twist.linear.y = 0.0;
                ↪  twist.linear.z = 0.0
198             twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z =
                ↪  self.target_angular_vel
```

```
199         self.pub.publish(twist)
200         rate.sleep()
201
202 if __name__ == "__main__":
203     teleop = KeyboardTeleop()
204     teleop.run()
```

Нижний уровень, файл автоматического управления и приема данных с датчиков. *Machine.ino*

```
1  #include <Wire.h>
2  #include <VL6180X.h>
3  #include <TroykaMQ.h>
4  #include "ros.h"
5  #include "std_msgs/String.h"
6  #include "geometry_msgs/Twist.h"
7
8  #define LOX1_ADDRESS 0x30
9  #define LOX2_ADDRESS 0x31
10 #define LOX3_ADDRESS 0x32
11 #define LOX4_ADDRESS 0x35
12 #define LOX5_ADDRESS 0x34
13 #define LOX6_ADDRESS 0x33
14
15 #define PIN_MQ6 A0
16
17 // set the pins to shutdown
18 #define SHT_LOX1 2
19 #define SHT_LOX2 3
20 #define SHT_LOX3 4
21 #define SHT_LOX4 7
22 #define SHT_LOX5 6
23 #define SHT_LOX6 5
24
25 #define BUF_LEN 128
26
27 //#define SERIAL_DEBUG
28 #define ROS
29
30 // lox1 - back-right
31 // lox2 - back-center
32 // lox3 - back-left
33 // lox4 - forward-right
34 // lox5 - forward-center
35 // lox6 - forward-left
36 VL6180X lox1, lox2, lox3, lox4, lox5, lox6;
37
38 // object for MQ-6:
39 MQ6 mq6(PIN_MQ6);
40
41 // for motors
42 const int motor1b_dir = 9;
43 const int motor1a_pwm = 8;
44 const int motor2b_dir = 11;
45 const int motor2a_pwm = 10;
46
47 // --- Ros Handler Init --- //
48 ros::NodeHandle NodeHandler;
49 std_msgs::String SensorsMessage;
50 ros::Publisher pub("SensorsData", &SensorsMessage);
```

```
51
52  // Message stream for ROS
53  char msg[BUF_LEN];
54  String s_msg;
55
56  //================================================================
57  // ROS
58  //================================================================
59
60  void getROScmdVel(const geometry_msgs::Twist& msg) {
61    if (msg.linear.x == 0 && msg.angular.z == 0) {
62      setStopMoving();
63      return;
64    }
65
66    int ispeed = choiceSpeed(msg.linear.x);
67    int lowerSpeed = choiceLowerSpeed(msg.angular.z, 1);
68    if (msg.linear.x > 0) {
69      if (msg.angular.z > 0) {
70        setTurningMoving(ispeed, lowerSpeed, HIGH);
71        //setLeftMoving();
72      }
73      else {
74        if (msg.angular.z < 0) {
75        setTurningMoving(lowerSpeed, ispeed, HIGH);
76        //setRightMoving();
77        }
78        else {
79          setForwardMoving(ispeed);
80        }
81      }
82    }
83    else {
84      lowerSpeed = choiceLowerSpeed(msg.angular.z, -1);
85
86      setBackMoving(ispeed);
87    }
88  }
89
90  int choiceSpeed(float value) {
91    if (value == 0 || abs(value) > 1) {
92      return 0;
93    }
94    if (abs(value) == 0.1) {
95      return value < 0 ? 1 : 255;
96    }
97    if (abs(value) == 0.2 || abs(value) == 0.3) {
98      return value < 0 ? 75 : 175;
99    }
100   if (abs(value) == 0.4) {
101     return value < 0 ? 100 : 150;
102   }
103   if (abs(value) == 0.5) {
104     return 125;
105   }
106   if (abs(value) == 0.6) {
107     return value < 0 ? 150 : 100;
108   }
109   if (abs(value) == 0.7) {
110     return value < 0 ? 175 : 75;
```

```
111     }
112     if (abs(value) == 0.8) {
113       return value < 0 ? 200 : 50;
114     }
115     if (abs(value) == 0.9) {
116       return value < 0 ? 225 : 25;
117     }
118     if (abs(value) == 1) {
119       return value < 0 ? 255 : 1;
120     }
121   }
122
123   int choiceLowerSpeed(float value, int direction) {
124     // ================== 1 ==================
125     if (value == 0 || abs(value) > 1) {
126       return 0;
127     }
128     value = abs(value);
129     if (value == 0.1 || value == 0.2 || value == 0.3 || value == 0.4 || value ==
        ↪  0.5) {
130       return direction < 0 ? 75 : 175;
131     }
132     if (value == 0.6 || value == 0.7) {
133       return direction < 0 ? 100 : 150;
134     }
135     if (value == 0.8 || value == 0.9) {
136       return direction < 0 ? 125 : 125;
137     }
138     if (value == 1) {
139       return direction < 0 ? 150 : 100;
140     }
141   }
142
143   void getROScmdVel2(const geometry_msgs::Twist& msg) {
144     float right_vel = 0.0;
145     float left_vel = 0.0;
146     float vel_x = 0.0;
147     float vel_th = 0.0;
148
149     if (msg.linear.x > 0.22) {
150       vel_x = 0.22;
151     }
152     else if (msg.linear.x < -0.22) {
153       vel_x = -0.22;
154     }
155     else {
156       vel_x = msg.linear.x;
157     }
158
159     if (msg.angular.z > 0.22) {
160       vel_th = 0.22;
161     }
162     else if (msg.angular.z < -0.22) {
163       vel_th = -0.22;
164     }
165     else {
166       vel_th = msg.angular.z;
167     }
168
169     if (vel_x == 0) {
```

```
    right_vel = vel_th * 0.16 / 2.0;
    left_vel = (-1) * right_vel;
  }
  else if (vel_th == 0) {
    left_vel = right_vel = vel_x;
  }
  else {
    left_vel = vel_x - vel_th / 2.0;
    right_vel = vel_x + vel_th / 2.0;
  }

  float RPMleft = ((60 * left_vel) / (0.07 * 3.1416));
  float RPMright = ((60 * right_vel) / (0.07 * 3.1416));

  int PWMleft = RPMleft * 4.25;
  int PWMright = RPMright * 4.25;

  PWMleft = PWMleft > 255 ? 255 : PWMleft;
  PWMright = PWMright > 255 ? 255 : PWMright;
  PWMleft = PWMleft < -255 ? -255 : PWMleft;
  PWMright = PWMright < -255 ? -255 : PWMright;

  int modeL = HIGH;
  if (PWMleft >= 0) {
    PWMleft = map(PWMleft, 0, 255, 255, 1);
  }
  else {
    modeL = LOW;
    PWMleft = abs(PWMleft);
  }

  int modeR = HIGH;
  if (PWMright >= 0) {
    PWMright = map(PWMleft, 0, 255, 255, 1);
  }
  else {
    modeR = LOW;
    PWMright = abs(PWMleft);
  }

  setMoving(PWMleft, PWMright, modeL, modeR);
}

ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel", getROScmdVel);

//================================================================
// Sensors
//================================================================

void setID() {
  // all reset
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  digitalWrite(SHT_LOX6, LOW);
  digitalWrite(SHT_LOX5, LOW);
  digitalWrite(SHT_LOX4, LOW);
  delay(10);

  // all unreset
```

```
230     digitalWrite(SHT_LOX1, HIGH);
231     digitalWrite(SHT_LOX2, HIGH);
232     digitalWrite(SHT_LOX3, HIGH);
233     digitalWrite(SHT_LOX6, HIGH);
234     digitalWrite(SHT_LOX5, HIGH);
235     digitalWrite(SHT_LOX4, HIGH);
236     delay(10);
237 
238     // activating LOX1 and reseting LOX2
239     digitalWrite(SHT_LOX1, HIGH);
240     digitalWrite(SHT_LOX2, LOW);
241     digitalWrite(SHT_LOX3, LOW);
242     digitalWrite(SHT_LOX6, LOW);
243     digitalWrite(SHT_LOX5, LOW);
244     digitalWrite(SHT_LOX4, LOW);
245     delay(10);
246 
247     lox1.init();
248     lox1.configureDefault();
249     lox1.setScaling(2);
250     lox1.setAddress(LOX1_ADDRESS);
251 
252     digitalWrite(SHT_LOX2, HIGH);
253     delay(10);
254 
255     lox2.init();
256     lox2.configureDefault();
257     lox2.setScaling(2);
258     lox2.setAddress(LOX2_ADDRESS);
259 
260     digitalWrite(SHT_LOX3, HIGH);
261     delay(10);
262 
263     lox3.init();
264     lox3.configureDefault();
265     lox3.setScaling(2);
266     lox3.setAddress(LOX3_ADDRESS);
267 
268     digitalWrite(SHT_LOX4, HIGH);
269     delay(10);
270 
271     lox4.init();
272     lox4.configureDefault();
273     lox4.setScaling(2);
274     lox4.setAddress(LOX4_ADDRESS);
275 
276     digitalWrite(SHT_LOX5, HIGH);
277     delay(10);
278 
279     lox5.init();
280     lox5.configureDefault();
281     lox5.setScaling(2);
282     lox5.setAddress(LOX5_ADDRESS);
283     delay(10);
284 
285     digitalWrite(SHT_LOX6, HIGH);
286     delay(10);
287 
288     lox6.init();
289     lox6.configureDefault();
```

```
290   lox6.setScaling(2);
291   lox6.setAddress(LOX6_ADDRESS);
292 }
293
294 void readSensors() {
295   s_msg = "";
296
297   s_msg.concat(lox1.readRangeSingleMillimeters());
298   s_msg.concat(";");
299   s_msg.concat(lox2.readRangeSingleMillimeters());
300   s_msg.concat(";");
301   s_msg.concat(lox3.readRangeSingleMillimeters());
302   s_msg.concat(";");
303   s_msg.concat(lox4.readRangeSingleMillimeters());
304   s_msg.concat(";");
305   s_msg.concat(lox5.readRangeSingleMillimeters());
306   s_msg.concat(";");
307   s_msg.concat(lox6.readRangeSingleMillimeters());
308   s_msg.concat(";");
309   s_msg.concat(mq6.readLPG());
310 }
311
312 //================================================================
313 // Motors
314 //================================================================
315
316 void setForwardMoving() {
317   digitalWrite(motor1b_dir, HIGH);
318   analogWrite(motor1a_pwm, 1);
319   digitalWrite(motor2b_dir, HIGH);
320   analogWrite(motor2a_pwm, 1);
321 }
322
323 void setBackMoving() {
324   digitalWrite(motor1b_dir, LOW);
325   analogWrite(motor1a_pwm, 255);
326   digitalWrite(motor2b_dir, LOW);
327   analogWrite(motor2a_pwm, 255);
328 }
329
330 void setLeftMoving() {
331   digitalWrite(motor1b_dir, HIGH);
332   analogWrite(motor1a_pwm, 1);
333   digitalWrite(motor2b_dir, LOW);
334   analogWrite(motor2a_pwm, 1);
335 }
336
337 void setRightMoving() {
338   digitalWrite(motor1b_dir, LOW);
339   //analogWrite(motor1a_pwm, 255);
340   analogWrite(motor1a_pwm, 1);
341   digitalWrite(motor2b_dir, HIGH);
342   analogWrite(motor2a_pwm, 1);
343 }
344
345 void setForwardMoving(int ispeed) {
346   digitalWrite(motor1b_dir, HIGH);
347   analogWrite(motor1a_pwm, ispeed);
348   digitalWrite(motor2b_dir, HIGH);
349   analogWrite(motor2a_pwm, ispeed);
```

```
350 }
351
352 void setBackMoving(int ispeed) {
353   digitalWrite(motor1b_dir, LOW);
354   analogWrite(motor1a_pwm, ispeed);
355   digitalWrite(motor2b_dir, LOW);
356   analogWrite(motor2a_pwm, ispeed);
357 }
358
359 void setTurningMoving(int ispeedL, int ispeedR, int mode) {
360   digitalWrite(motor1b_dir, mode);
361   analogWrite(motor1a_pwm, ispeedL);
362   digitalWrite(motor2b_dir, mode);
363   analogWrite(motor2a_pwm, ispeedR);
364 }
365
366 void setMoving(int ispeedL, int ispeedR, int modeL, int modeR) {
367   digitalWrite(motor1b_dir, modeL);
368   analogWrite(motor1a_pwm, ispeedL);
369   digitalWrite(motor2b_dir, modeR);
370   analogWrite(motor2a_pwm, ispeedR);
371 }
372
373 void setStopMoving() {
374   digitalWrite(motor1b_dir, LOW);
375   analogWrite(motor1a_pwm, 1);
376   digitalWrite(motor2b_dir, LOW);
377   analogWrite(motor2a_pwm, 1);
378 }
379
380 //=================================================================
381 // Setup
382 //=================================================================
383 void setup() {
384   Serial.begin(115200);
385   Wire.begin();
386
387   pinMode(SHT_LOX1, OUTPUT);
388   pinMode(SHT_LOX2, OUTPUT);
389   pinMode(SHT_LOX3, OUTPUT);
390   pinMode(SHT_LOX6, OUTPUT);
391   pinMode(SHT_LOX5, OUTPUT);
392   pinMode(SHT_LOX4, OUTPUT);
393
394   pinMode(motor1b_dir, OUTPUT);
395   pinMode(motor1a_pwm, OUTPUT);
396   pinMode(motor2b_dir, OUTPUT);
397   pinMode(motor2a_pwm, OUTPUT);
398
399   setID();
400
401   #ifdef SERIAL_DEBUG
402     Serial.println("Starting...");
403   #endif
404
405   // Init ROS Nodes
406   #ifdef ROS
407     NodeHandler.getHardware()->setBaud(115200);
408     NodeHandler.initNode();
409     NodeHandler.advertise(pub);
```

```
410       NodeHandler.subscribe(sub);
411     #endif
412
413     delay(60000);
414     mq6.calibrate();
415
416     #ifdef SERIAL_DEBUG
417       Serial.print("Ro: ");
418       Serial.println(mq6.getRo());
419     #endif
420   }
421
422   //================================================================
423   // Loop
424   //================================================================
425   void loop() {
426
427     readSensors();
428
429     memset(msg, '\0', sizeof(msg));
430     s_msg.toCharArray(msg, BUF_LEN);
431     SensorsMessage.data = msg;
432
433     #ifdef SERIAL_DEBUG
434       Serial.print("Ratio: ");
435       Serial.print(mq6.getRo());
436       Serial.print(" ; LPG: ");
437       Serial.println(mq6.readLPG());
438       Serial.println(SensorsMessage.data);
439     #endif
440
441     #ifdef ROS
442       pub.publish(&SensorsMessage);
443       NodeHandler.spinOnce();
444     #endif
445
446     delay(125);
447   }
```

Конструкционная часть