Беспилотные авиационные системы

Заключительный этап

Инженерный тур

Общая информация

Мониторинг промышленного объекта в автоматическом режиме с помощью БПЛА самолетного типа и анализе полученных данных в режиме реального времени с помощью симулятора виртуального полета БПЛА.

Сюжет задачи

Моделируется ситуация техногенной катастрофы — например, взрыв на химическом заводе при землетрясении. При этом образовались выбросы продуктов химических реакций (зона загрязнения). В результате этих выбросов образовалось облако (зона) загрязнений.

Задача — оперативно в автоматическом режиме с применением БПЛА самолетного типа получить информацию об этой зоне (размер, концентрация вредных веществ). При этом сначала моделируется ситуация, при которой облако загрязнений переместилось на некоторое расстояние, а затем зависло (ветра нет) и выпала часть осадков.

Задача БПЛА — не только собрать информация об этой зоне, но и автоматически по полученному снимку определить геометрию зоны осадков.

Участники на финале работают не только с симулятором полета, но и с реальными датчиками, органами управления и двигательной установкой БПЛА самолетного типа.

Требования к команде и компетенциям участников

Количество участников в команде: 4.

Компетенции, которыми должны обладать члены команды:

(роли, которые должны быть представлены в команде)

1. Капитан (взаимодействие со всеми участниками команды, формирование окончательных ответов и принятие решений).
2. Математик (математическая обработка сигналов с датчиков, формирование траектории полёта, разработка алгоритмов).
3. Электронщик (работа с датчиками и органами управления БПЛА — подключение и тестирование).
4. Программист (реализация алгоритмов решения задач).

Оборудование и программное обеспечение

Наименование	Описание
Микромеханические инерциальные датчики (акселерометры и гироскопы)	Используются для разработки алгоритмов определения углов ориентации БПЛА
Датчики влажности воздуха	Используются в качестве имитации бортовых газоанализаторов при разработке алгоритма определения параметров зоны загрязнения
БПЛА Skywalker 2015	Используется для отработки алгоритмов управления БПЛА
Языки программирования C++ и Python	Используются для написания программного кода во всех задачах финала
Симулятор полёта БПЛА UAviant	Используется для отработки решений участников в виртуальной среде

Задача VI.2.4.1.

Для обеспечения управляемого полёта БПЛА оснащён множеством систем (органов) управления:

- руль высоты для разворота по тангажу;
- руль поворота для осуществления разворота по курсу;
- элероны для разворота по углу крена;
- двигатель для изменения скорости полёта.

Для эффективного управления полётом требуется одновременное управление всеми этими системами.

Условие

Ознакомьтесь с органами управления летательного аппарата: двигателем, элеронами, рулём высоты и рулём направления. Напишите программу на Arduino, реализующую переход БПЛА в заданный режим полёта поочередно (исполнение заданного режима должно выполнятся в течении 5 с).

Требуется реализовать:

- управление двигателем для увеличения путевой скорости БПЛА;
- управление элеронами для разворота по курсу вправо;
- управление рулём высоты для снижения БПЛА;
- управление рулём направления для разворота по курсу влево.

Формат входных данных

Плата Arduino и шаблон программы.

Формат выходных данных

Программа на Arduino, реализующая четыре заданных режима полёта БПЛА поочередно.

Критерии оценивания

1. Факт достижения БПЛА заданных режимов полёта оценивается организаторами на полунатурном стенде.
2. За успешную реализацию каждого из режимов начисляется 1 балл.
3. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Настроить режим вывода на пинах органов управления.
2. Создать функции перевода параметров управления из СИ в ШИМ.
3. Задать требуемые сигналы управления.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include "Task1.h"

void Task1_Solution::init()
{
  pinMode(ALI_left, OUTPUT);
  pinMode(ALI_right, OUTPUT);
  pinMode(ELE, OUTPUT);
  pinMode(THR, OUTPUT);
  pinMode(RUD, OUTPUT);
  aleron_l.attach(ALI_left);
  aleron_r.attach(ALI_right);
  elevator.attach(ELE);
  throttle.attach(THR);
  rudder.attach(RUD);
  throttle.write(0);
  aleron_l.write(map(0, ELERONS_DEG_MIN, ELERONS_DEG_MAX, ELERONS_PWM_MIN,
  ↪  ELERONS_PWM_MAX));
  aleron_r.write(map(0, ELERONS_DEG_MIN, ELERONS_DEG_MAX, ELERONS_PWM_MIN,
  ↪  ELERONS_PWM_MAX));
  rudder.write(map(0, RUDDER_DEG_MIN, RUDDER_DEG_MAX, RUDDER_PWM_MIN,
  ↪  RUDDER_PWM_MAX));
  elevator.write(map(0, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX, ELEVATOR_PWM_MIN,
  ↪  ELEVATOR_PWM_MAX));
}

void Task1_Solution::servo_out(Task1_PWM a_pwm)
{
  elevator.write(a_pwm.Elevator_PWM);
  rudder.write(a_pwm.Rudder_PWM);
  // Элероны
  aleron_l.write(a_pwm.Elerons_PWM);
  aleron_r.write(a_pwm.Elerons_PWM);
```

```
29    // Движок
30    throttle.write(a_pwm.Engine_PWM);
31  }
32
33  Task1_PWM Task1_Solution::Task1_in_the_loop(const float& a_Elevator_zad, const
    ↪  float& a_Rudder_zad, const float& a_Elerons_zad, const float& a_Thrust_zad)
34  {
35    Task1_PWM _ans;
36    _ans.Elerons_PWM = map(a_Elerons_zad, ELERONS_DEG_MIN, ELERONS_DEG_MAX,
      ↪  ELERONS_PWM_MIN, ELERONS_PWM_MAX);
37    _ans.Rudder_PWM = map(a_Rudder_zad, RUDDER_DEG_MIN, RUDDER_DEG_MAX,
      ↪  RUDDER_PWM_MIN, RUDDER_PWM_MAX);
38    _ans.Elevator_PWM = map(a_Elevator_zad, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX,
      ↪  ELEVATOR_PWM_MIN, ELEVATOR_PWM_MAX);
39    _ans.Engine_PWM = map(a_Thrust_zad, RPM_MIN, RPM_MAX, RPM_PWM_MIN, RPM_PWM_MAX);
40    servo_out(_ans);
41    return _ans;
42  }
43
44  Task1_PWM Task1_Solution::Task1_servo_thurst_V()
45  {
46    Task1_PWM _ans;
47    _ans.Elerons_PWM = map(0, ELERONS_DEG_MIN, ELERONS_DEG_MAX,ELERONS_PWM_MIN,
      ↪  ELERONS_PWM_MAX);
48    _ans.Rudder_PWM = map(0, RUDDER_DEG_MIN, RUDDER_DEG_MAX,RUDDER_PWM_MIN,
      ↪  RUDDER_PWM_MAX);
49    _ans.Elevator_PWM = map(0, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX,ELEVATOR_PWM_MIN,
      ↪  ELEVATOR_PWM_MAX);
50    _ans.Engine_PWM = map(10, RPM_MIN, RPM_MAX,RPM_PWM_MIN, RPM_PWM_MAX);
51
52    //
53    servo_out(_ans);
54    return _ans;
55  }
56
57  Task1_PWM Task1_Solution::Task1_servo_Elerons_R()
58  {
59    Task1_PWM _ans;
60    _ans.Elerons_PWM = map(20, ELERONS_DEG_MIN, ELERONS_DEG_MAX, ELERONS_PWM_MIN,
      ↪  ELERONS_PWM_MAX);
61    _ans.Rudder_PWM = map(0, RUDDER_DEG_MIN, RUDDER_DEG_MAX, RUDDER_PWM_MIN,
      ↪  RUDDER_PWM_MAX);
62    _ans.Elevator_PWM = map(0, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX, ELEVATOR_PWM_MIN,
      ↪  ELEVATOR_PWM_MAX);
63    _ans.Engine_PWM = map(10, RPM_MIN, RPM_MAX, RPM_PWM_MIN, RPM_PWM_MAX);
64    servo_out(_ans);
65    return _ans;
66  }
67
68  Task1_PWM Task1_Solution::Task1_servo_Elevator()
69  {
70    Task1_PWM _ans;
71    _ans.Elerons_PWM = map(0, ELERONS_DEG_MIN, ELERONS_DEG_MAX, ELERONS_PWM_MIN,
      ↪  ELERONS_PWM_MAX);
72    _ans.Rudder_PWM = map(0, RUDDER_DEG_MIN, RUDDER_DEG_MAX, RUDDER_PWM_MIN,
      ↪  RUDDER_PWM_MAX);
73    _ans.Elevator_PWM = map(15, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX,
      ↪  ELEVATOR_PWM_MIN, ELEVATOR_PWM_MAX);
74    _ans.Engine_PWM = map(10, RPM_MIN, RPM_MAX, RPM_PWM_MIN, RPM_PWM_MAX);
75    servo_out(_ans);
```

```
76    return _ans;
77  }
78
79  Task1_PWM Task1_Solution::Task1_servo_Rudder()
80  {
81    Task1_PWM _ans;
82    _ans.Elerons_PWM = map(0, ELERONS_DEG_MIN, ELERONS_DEG_MAX, ELERONS_PWM_MIN,
      ↪  ELERONS_PWM_MAX);
83    _ans.Rudder_PWM = map(-10, RUDDER_DEG_MIN, RUDDER_DEG_MAX, RUDDER_PWM_MIN,
      ↪  RUDDER_PWM_MAX);
84    _ans.Elevator_PWM = map(0, ELEVATOR_DEG_MIN, ELEVATOR_DEG_MAX, ELEVATOR_PWM_MIN,
      ↪  ELEVATOR_PWM_MAX);
85    _ans.Engine_PWM = map(10, RPM_MIN, RPM_MAX, RPM_PWM_MIN, RPM_PWM_MAX);
86    servo_out(_ans);
87    return _ans;
88  }
```

Материалы для подготовки

- ШИМ сигнал в Arduino: https://alexgyver.ru/lessons/pwm-signal/.

Задача VI.2.4.2.

Для реализации алгоритмов автоматического управления полётом БПЛА необходимо как можно точнее определять его навигационные параметры в каждый момент времени.

В современных бортовых измерительных системах используются алгоритмы комплексной обработки информации, позволяющие совместно обрабатывать измерения разных датчиков для повышения точности навигационного решения.

Условие

Напишите программу для Arduino, реализующую расчёт угла тангажа по измерениям акселерометра и гироскопа с использованием комплементарной фильтрации. Для повышения точности определения угла тангажа необходимо провести начальную калибровку датчиков.

Формат входных данных

Датчик MPU6050 и плата Arduino.

Формат выходных данных

Программа на Arduino, реализующая расчёт угла тангажа по измерениям гироскопа и акселерометра.

Критерии оценивания

1. Для оценки точности расчёта угла датчик последовательно поворачивается на углы 30, 45, 60°. Оценивается средняя ошибка определения угла тангажа.

2. Если величина ошибки составляет меньше 2° за решение задачи начисляется 5 баллов.
3. За каждый дополнительный 1 полный градус ошибки снимается 1 балл.
4. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Провести калибровку датчика и определить коэффициенты.
2. Скорректировать измерения датчиков.
3. Рассчитать угол тангажа.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
    delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  Serial.println("");
  delay(100);
}

float gammag = 0.0;
float gammaa = 0.0;
float acc[] = {0.2, 0.2, 0.08};
float gyr[] = {0.02, 0, 0.01};
float acc_scale = 1.03;

void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  float gyr_x = g.gyro.x + gyr[0];
  float acc_z = (a.acceleration.z + acc[2]) * acc_scale;
```

```
42    float acc_y = (a.acceleration.y + acc[1]) * acc_scale;
43    gammaa = atan2(acc_y, acc_z) * 180.0 / 3.1415;
44    if (abs(gyr_x) > 0.07)
45    {
46      gammag += gyr_x * 0.2 * 180.0 / 3.1415;
47    }
48    float k = 0.3;
49    float angle = k * gammag + (1 - k) * gammaa;
50    Serial.print("Angle: ");
51    Serial.print(gammaa);
52    Serial.print(" + ");
53    Serial.print(gammag);
54    Serial.print(" = ");
55    Serial.println(angle);
56    delay(200);
57  }
```

Материалы для подготовки

- MPU6050 Arduino: https://alexgyver.ru/arduino-mpu6050/.

Задача VI.2.4.3.

Одним из перспективных применений БПЛА является автоматический мониторинг экологической обстановки. Например, БПЛА может быть оснащён газоанализатором, позволяющим ему анализировать качество окружающего воздуха.

Оперативное определение экологической обстановки в удалённых областях позволит обеспечить рациональное и экологически эффективное использование природных пространств.

Условие

Задача делится на две части.

В первой части необходимо разработать программу на Arduino, реализующую анализ измерений датчика влажности DHT22 и зажигающую светодиод при превышении влажности порогового значения.

Во второй части задачи участникам необходимо написать программу на Python, реализующую обмен данными с симулятором и расчёт радиуса зоны атмосферного загрязнения.

Формат входных данных

Симулятор, датчик DHT22 и плата Arduino.

Формат выходных данных

Программа на Arduino, реализующая анализ измерений датчика влажности, и программа на Python, реализующая расчёт радиуса зоны атмосферного загрязнения.

Критерии оценивания

Каждая часть задания оценивается отдельно.

1. Датчик DHT22, подключённый к Arduino, перемещается сквозь область повышенной влажности. Если светодиод загорается — часть задачи считается выполненной и за неё начисляется 1 балла.
2. Программа расчёта радиуса зоны загрязнения проверяется на симуляторе. Если ошибка определения радиуса не превышает 10 м, за решение задачи начисляется 3 балла. Если ошибка в пределах 20 м — 2 балла. Если в пределах 30 м — 1 балл.
3. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Подключить датчик.
2. Подобрать порог включения.
3. Разработать алгоритм определения координат входа и выхода БПЛА из облака.
4. Рассчитать радиус зоны загрязнения.

Пример программы-решения

Ниже представлено решение на Arduino.

```
#include "DHT.h"

#define DHTPIN 2     // Digital pin connected to the DHT sensor

// Uncomment whatever type you're using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(250);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();

  if (h < 40)
  {
    digitalWrite(13, LOW);
  }
  else
```

```
  {
    digitalWrite(13, HIGH);
  }

  // Check if any reads failed and exit early (to try again).
  if (isnan(h)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.println("%");
}
```

Ниже представлено решение на языке Python 3.

```
# Поток обработки данных с датчика
def sensor_processing_thread():
    x1 = 0
    z1 = 0
    x2 = 0
    z2 = 0
    last_measurement = 0
    print("ready")
    r = []
    # Основной цикл программы
    while running:
        tele = last_telemetry
        if tele.air != 0 and last_measurement == 0:
            x1 = tele.x
            z1 = tele.z
            print("in")
        if tele.air == 0 and last_measurement != 0:
            x2 = tele.x
            z2 = tele.z
            print("out")
            r.append(sqrt((x2-x1)**2 + (z2 - z1)**2) / 2)
            print(f"Radius: {r[-1]}, {sum(r)/len(r)}")

        last_measurement = tele.air
        time.sleep(0.02)
```

Материалы для подготовки

- Подключение датчика DHT22 к Arduino: https://geekelectronics.org/arduino/dht22-podklyuchenie-k-arduino.html.

Задача VI.2.4.4.

Одной из базовых функций, реализуемых БПЛА, является автономный полёт по заданной траектории. Для осуществления такого полёта разрабатываются специальные системы автоматического управления, способные регулировать все параметры навигации БПЛА в зависимости от его положения относительно заданной траектории.

Условие

Разработайте программу, реализующую автоматическое управление углом крена и тангажа БПЛА, для осуществления его движения по заданной траектории. Траектория движения формируется участниками команды произвольным образом с целью пролёта БПЛА по расположенным в симуляторе путевым точкам.

Формат входных данных

Координаты путевых точек, система автоматического управления БПЛА (кроме регуляторов в каналах крена и тангажа); навигационные параметры БПЛА в каждый момент времени (координаты и углы ориентации).

Формат выходных данных

Программа на Arduino, реализующая автономный полёт БПЛА по заданным путевым точкам.

Критерии оценивания

В симуляторе расположены 10 путевых точек в виде цветных сфер диаметром 15 м. Оценивается количество и тип пройденных БПЛА путевых точек.

1. За каждую пройденную красную путевую точку начисляется 1 балл.
2. За каждую пройденную фиолетовую путевую точку начисляется 2 балла.
3. Время полёта БПЛА не должно превышать 5 мин.
4. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Разработать регулятор канала высоты.
2. Разработать регулятор канала крена.
3. Выбрать оптимальную траекторию облёта точек.
4. Проверить на симуляторе и скорректировать коэффициенты управления.

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  float UARTControlSystemClass::HeightReg(const Skywalker2015PacketTelemetry& _tele,
   ↪  const float &Hz) // _tele -  структура принятого пакета телеметрии, Hz -
   ↪  заданная высота
2  {
3    float K_H = 0.56;
4    float K_vy_1 = 7.7591;
5    float K_vy_2 = 9.8941;
6    float Yg = _tele.H; //  Получаем из принятого пакета телеметрии текущую высоту
     ↪  БПЛА, м.
7    float Vy = _tele.Vy1; // Получаем из принятого пакета телеметрии текущую
     ↪  вертикальную скорость БПЛА, м/с.
```

```
8
9     float kh1;
10    float _Pitch_direct;
11    float hz_corr = _New_Waypoint.H - 2;
12    kh1 = Saturation(K_H * (hz_corr - Yg), -7.5, 7.5);  // Интегральное звено +
      ↪  Ограничитель
13    _Pitch_direct = Saturation((kh1 * K_vy_1 - Vy * K_vy_2), -20, 20); //
      ↪  Дифференциальное звено + Ограничитель
14
15    return _Pitch_direct; // Возвращаем заданный тангаж.
16  }
17
18  float UARTControlSystemClass::ToPointXY(const Skywalker2015PacketTelemetry& _tele,
    ↪  const float& a_Xg, const float& a_Yg)
19  {
20    float _Xt = _tele.L;
21    float _Yt = _tele.Z;
22
23    float xg_corr = a_Xg;
24    float yg_corr = a_Yg;
25
26    float Psi_cmd = -atan2(yg_corr - _Yt, xg_corr - _Xt);//% курс на точку, радианы
27    Psi_cmd *= 57.3; // Переведем в градусы
28    return GammaReg(_tele, Psi_cmd);
29  }
30
31  float UARTControlSystemClass::GammaReg(const Skywalker2015PacketTelemetry& _tele,
    ↪  const float& Psi_direct_deg)
32  {
33    static bool _doonce = true;
34    static bool _doonce2 = true;
35    static float Psi_direct;
36    float gamma_cmd;
37    float Psi_dir = Psi_direct_deg;
38    float Xg_cmd = _PointsArray[_Point_Index].L;
39    float Zg_cmd = _PointsArray[_Point_Index].Z;
40    float Yg_cmd = _PointsArray[_Point_Index].H;
41    float delX = Xg_cmd - _tele.L;
42    float delY = Zg_cmd - _tele.Z;
43    float delZ = Yg_cmd - _tele.H;
44
45    float Size = sqrtf(pow(delX, 2) + pow(delY, 2) + pow(delZ, 2));
46    // Регулятор курса
47    float Kpsi = 1.6;
48    float Psi = _tele.Psi; // Получаем с принятого пакета телеметрии текущий угол
      ↪  курса(в связанной СК), град.
49
50    gamma_cmd = Kpsi * (AngDefines(Psi - Psi_dir));
51    gamma_cmd = Saturation(gamma_cmd, -40, 40);
52
53    return gamma_cmd; // Заданный угол крена, град
54  }
55
56  size_t UARTControlSystemClass::GetNowPointIndex(const Skywalker2015PacketTelemetry
    ↪  & _tele)
57  {
58    size_t max_index = 15;
59    //SendTele(_tele);
60    float Xg_cmd = _PointsArray[_Point_Index].L;
61    float Zg_cmd = _PointsArray[_Point_Index].Z;
```

```
  float Yg_cmd = _PointsArray[_Point_Index].H;
  if (_User_Control_Enabled)
  {
    Xg_cmd =  _New_Waypoint.L;
    Zg_cmd =  _New_Waypoint.Z;
    Yg_cmd =  _New_Waypoint.H;
  }
  float delX = Xg_cmd - _tele.L;
  float delY = Zg_cmd - _tele.Z;
  float delZ = Yg_cmd - _tele.H;
  float Size = sqrtf(pow(delX, 2) + pow(delY, 2) + pow(delZ, 2));
  // Calculate crosstrack error
  float x_og = 0.0;
  float z_og = 0.0;
  if (_Point_Index > 0)
  {
    x_og = _PointsArray[_Point_Index - 1].L;
    z_og = _PointsArray[_Point_Index - 1].Z;
  }
  else
  {
    x_og = _PointsArray[max_index - 1].L;
    z_og = _PointsArray[max_index - 1].Z;
  }
  float relx = _tele.L - x_og;
  float rely = _tele.Z - z_og;
  float delx = Xg_cmd - x_og;
  float dely = Zg_cmd - z_og;
  float U = (relx * delx + rely * dely) / (delx * delx + dely * dely);
  float cte = (rely * delx - relx * dely) / (delx * delx + dely * dely);
  _Trajectory_Offset = cte * 1500.0;
  if (Size < 6.0)
  {
    if (_User_Control_Enabled)
    {
      _User_Control_Enabled = false;
      //Serial3.println("contol disabled");
    }
    else
    {
    _Point_Index++;
    }
  }
  // Выход за границы массива
  if (_Point_Index >= max_index)
  {
    _Point_Index = 0;
  }
  return _Point_Index;
}
```

Материалы для подготовки

- Советы по настройке ПИД-регулятора: https://realpars.com/pid-tuning/.

Задача VI.2.4.5.

При выполнении миссий автоматического экологического мониторинга БПЛА должен иметь возможность определять параметры окружающей среды с учётом информации, получаемой от всех бортовых измерительных систем.

В частности, полезность информации о низком качестве окружающего воздуха значительно повышается, если БПЛА может определить координаты точки, в которой было получено это измерение.

Условие

Разработайте программу на Python, реализующую автоматическое определение площади зоны загрязнения на основе анализа данных телеметрии БПЛА — его навигационных параметров и измерений бортового газоанализатора. Точное расположение зоны загрязнения в симуляторе неизвестно, однако, участники могут задавать путевые точки для системы автопилота для управления движением БПЛА.

Формат входных данных

Система автоматического управления БПЛА; навигационные параметры БПЛА в каждый момент времени (координаты и углы ориентации); измерения датчика качества воздуха.

Формат выходных данных

Программа на Python, реализующая автоматический расчёт площади зоны загрязнения.

Критерии оценивания

Оценивается точность определения площади зоны загрязнения.

1. Если величина ошибки составляет меньше 2000 м^2 за решение задачи начисляется 10 баллов.
2. За каждую дополнительную 1000 м^2 ошибки снимается 1 балл.
3. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Сформировать траекторию полёта БПЛА.
2. Определить координаты точек на границе зоны загрязнения.
3. Рассчитать площадь зоны загрязнения.
4. Проверить на симуляторе и скорректировать коэффициенты управления.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
def get_area(points_x, points_y):
    mx = mean(points_x)
    my = mean(points_y)
    arrx = []
    arry = []
    for i in range(0, len(points_x)):
        arrx.append(points_x[i] - mx)
        arry.append(points_y[i] - my)
    S = 0.5 * (arry[0]*arrx[-1] - arry[-1]*arrx[0]);
    for i in range(0, len(arrx) - 1):
        S += 0.5 * (arry[i]*arrx[i+1] - arry[i+1]*arrx[i])
    S = abs(S)
    print(f"Center approx: {mx} {my}")
    print("AREA: {:.1f}".format(S))
    print(len(arrx))

# Поток обработки данных с датчика
def sensor_processing_thread():
    points_x = []
    points_z = []
    last_measurement = 0
    targets = [(120, 150, 100), (100, 500, 100), (300, 400, 100), (-200, 300, 100)]
    cpt = targets
    count = 0
    dist = 0
    sendNewWaypoint(targets[0][0], targets[0][1], targets[0][2])
    r = []
    c = 0

    # Основной цикл программы
    while running:
        tele = last_telemetry
        dist = sqrt((targets[count][0] - tele.x)**2 + (targets[count][1] - tele.z)**2)
        if dist <= 10:
            count = (count + 1)
            if (count >= len(targets)):
                targets = []
                count = 0
                for i in range(0, len(cpt)):
                    targets.append((cpt[i][0] + random.uniform(-25, 25.0), cpt[i][1] + random.uniform(-25, 25.0), 100))
            sendNewWaypoint(targets[count][0], targets[count][1], targets[count][2])
        # Добавьте ваше решение сюда
        if last_measurement == 0 and tele.air != 0:
            points_x.append(tele.x)
            points_z.append(tele.z)
            #print(f"added {tele.x} {tele.z}")
            #get_area(points_x, points_z)
        if last_measurement != 0 and tele.air == 0:
            points_x.append(tele.x)
            points_z.append(tele.z)
            s = sqrt((points_x[-1] - points_x[-2])**2 +
                     (points_z[-1] - points_z[-2])**2)/2
            r.append(s)
            print(f"Mean radius {mean(r)} approx to {mean(r)**2*pi}")
            #print(f"added {tele.x} {tele.z}")
            #get_area(points_x, points_z)
```

```
57
58          last_measurement = tele.air
59          time.sleep(0.05)
```

Материалы для подготовки

- ШИМ сигнал в Arduino: https://alexgyver.ru/lessons/pwm-signal/.

Задача VI.2.4.6.

Режим автоматического следования за подвижным объектом является одним из основных режимов при реализации задач автоматической разведки и мониторинга.

При этом сложность системы автоматического управления БПЛА повышается, так как, в общем случае, скорость движения объекта отличается от скорости полёта БПЛА.

Условие

Разработайте программу на Arduino, реализующую автоматическое управление полётом БПЛА для осуществления его движения за подвижным объектом. БПЛА автоматически отслеживает местоположение объекта в каждый момент времени, однако скорость движения объекта ниже, чем скорость движения БПЛА.

Формат входных данных

Координаты объекта в каждый момент времени, система автоматического управления БПЛА (с задачи VI.2.4.4); навигационные параметры БПЛА в каждый момент времени (координаты и углы ориентации).

Формат выходных данных

Программа на Arduino, реализующая автономный полёт БПЛА за подвижным объектом.

Критерии оценивания

Время моделирования фиксированное — 90 с. Оценивается количество времени, которое БПЛА провёл в пределах 15 м от подвижного объекта.

1. Если БПЛА находился рядом с объектом 50 с или дольше — за решение задачи начисляется 14 баллов.
2. За каждые дополнительные 3 с когда БПЛА находился вдали от объекта снимается 1 балл.
3. Таким образом, для получения баллов за решение БПЛА должен находиться рядом с объектом, как минимум, 11 с.
4. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Доработать законы управления полётом БПЛА.
2. Определить скорость и направление движения объекта.
3. Сформировать траекторию следования за объектом «галсами».
4. Проверить на симуляторе.

Пример программы-решения

Ниже представлено решение на языке C++.

```
float x1 = 0;
float z1 = 0;
float x2 = 0;
float z2 = 0;
bool p1set = false;
bool p2set = false;

float UARTControlSystemClass::ToPointXY(const Skywalker2015PacketTelemetry& _tele,
↪  const float& a_Xg, const float& a_Yg)
{
  float _Xt = _tele.L;
  float _Yt = _tele.Z;

  if (!p2set && p1set)
  {
    x2 = _New_Waypoint.L;
    z2 = _New_Waypoint.Z;
    p2set = true;
    build_follow_traj(_Xt, _Yt);
  }
  if (!p1set)
  {
    x1 = _New_Waypoint.L;
    z1 = _New_Waypoint.Z;
    p1set = true;
  }
  float xg_corr = a_Xg;
  float yg_corr = a_Yg;
  float Psi_cmd = -atan2(yg_corr - _Yt, xg_corr - _Xt);//% курс на точку, радианы
  Psi_cmd *= 57.3; // Переведем в градусы
  return GammaReg(_tele, Psi_cmd);
}

void UARTControlSystemClass::build_follow_traj(float x, float z)
{
  float xvar = 0;
  float zvar = 0;
  float xdist = 0;
  float zdist = 0;
  if (abs(x2 - x1) < 0.1)
  {
    xvar = 10;
    zvar = 0;
    zdist = 300;
  }
  if (abs(z2 - z1) < 0.1)
  {
```

```
47      zvar = 10;
48      xvar = 0;
49      xdist = 300;
50    }
51    if (xdist > 0 && zdist> 0)
52    {
53      p2set = false;
54      return;
55    }
56    // Expanded trajectory
57    _PointsArray[0] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
      ↪  _New_Waypoint.H)); // wp
58     if (xdist > 0)
59      xdist = xdist + 100;
60     if (zdist > 0)
61      zdist = zdist + 100;
62    _PointsArray[1] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
      ↪  _New_Waypoint.H)); // wp
63     if (xdist > 0)
64      xdist = xdist + 100;
65     if (zdist > 0)
66      zdist = zdist + 100;
67    _PointsArray[2] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
      ↪  _New_Waypoint.H)); // wp
68     if (xdist > 0)
69      xdist = xdist + 100;
70     if (zdist > 0)
71      zdist = zdist + 100;
72    _PointsArray[3] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
      ↪  _New_Waypoint.H)); // wp
73     if (xdist > 0)
74      xdist = xdist + 100;
75     if (zdist > 0)
76      zdist = zdist + 100;
77    _PointsArray[4] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
      ↪  _New_Waypoint.H)); // wp
78     if (xdist > 0)
79      xdist = xdist + 100;
80     if (zdist > 0)
81      zdist = zdist + 100;
82    _PointsArray[5] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
      ↪  _New_Waypoint.H)); // wp
83     if (xdist > 0)
84      xdist = xdist + 100;
85     if (zdist > 0)
86      zdist = zdist + 100;
87    _PointsArray[6] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
      ↪  _New_Waypoint.H)); // wp
88     if (xdist > 0)
89      xdist = xdist + 100;
90     if (zdist > 0)
91      zdist = zdist + 100;
92    _PointsArray[7] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
      ↪  _New_Waypoint.H)); // wp
93     if (xdist > 0)
94      xdist = xdist + 100;
95     if (zdist > 0)
96      zdist = zdist + 100;
97    _PointsArray[8] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
      ↪  _New_Waypoint.H)); // wp
```

```
98      if (xdist > 0)
99        xdist = xdist + 100;
100      if (zdist > 0)
101       zdist = zdist + 100;
102     _PointsArray[9] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
        ↪  _New_Waypoint.H)); // wp
103     if (xdist > 0)
104       xdist = xdist + 100;
105      if (zdist > 0)
106       zdist = zdist + 100;
107     _PointsArray[10] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
        ↪  _New_Waypoint.H)); // wp
108     if (xdist > 0)
109       xdist = xdist + 100;
110      if (zdist > 0)
111       zdist = zdist + 100;
112     _PointsArray[11] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
        ↪  _New_Waypoint.H)); // wp
113     if (xdist > 0)
114       xdist = xdist + 100;
115      if (zdist > 0)
116       zdist = zdist + 100;
117     _PointsArray[12] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
        ↪  _New_Waypoint.H)); // wp
118     if (xdist > 0)
119       xdist = xdist + 100;
120      if (zdist > 0)
121       zdist = zdist + 100;
122     _PointsArray[13] = (FlyPoints(x2 + xdist + xvar, z2 + zdist + zvar,
        ↪  _New_Waypoint.H)); // wp
123     if (xdist > 0)
124       xdist = xdist + 100;
125      if (zdist > 0)
126       zdist = zdist + 100;
127     _PointsArray[14] = (FlyPoints(x2 + xdist - xvar, z2 + zdist - zvar,
        ↪  _New_Waypoint.H)); // wp
128   }
```

Задача VI.2.4.7.

Определение параметров различных объектов при помощи системы технического зрения является одной из самых распространённых задач, решаемых БПЛА. В частности, при выполнении мониторинга экологической обстановки может потребоваться определить площадь зоны осадков вредоносных веществ для оценки уровня экологической угрозы.

Условие

Разработайте программу на языке Python, реализующую расчёт площади поля осадков на статическом снимке с камеры при известных параметрах телеметрии БПЛА на момент съёмки.

Формат входных данных

Изображение с бортовой камеры БПЛА; параметры бортовой камеры; навигационные параметры БПЛА (координаты и углы ориентации) на момент осуществления съёмки.

Формат выходных данных

Программа реализующая расчёт площади поля осадков с использованием алгоритмов технического зрения и библиотеки OpenCV.

Критерии оценивания

Оценивается точность определения площади поля осадков.

1. Если величина ошибки составляет меньше 25 м^2 за решение задачи начисляется 10 баллов.
2. Если величина ошибки лежит в диапазоне 25–35 м^2 — начисляется 8 баллов.
3. Если величина ошибки лежит в диапазоне 35–50 м^2 — начисляется 6 баллов.
4. Если величина ошибки лежит в диапазоне 50–80 м^2 — начисляется 4 балла.
5. Если величина ошибки лежит в диапазоне 80–120 м^2 — начисляется 2 балла.
6. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Наложить цветовой фильтр на изображение.
2. Выделить контур искомой области.
3. Найти центр области.
4. Найти координаты каждой точки контура относительно центра области.
5. Рассчитать площадь.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import cv2 # used for image proccessing, duh
import numpy as np # used to convert images from PIL to cv2
import math # used for trigonometry functions

# Параметры БПЛА
uavX = 600 # degrees
uavY = 300 # degrees
uavYaw = 40 # degrees
uavPitch = 3 # degrees
H = 45 # meters
# Параметры камеры
dv = 0.077 # meters
dh = 0.09
width = 1000 # pixels
height = 770 # pixels
fieldHor = 90; # degrees
```

```
fieldVer = 60; # degrees

#color definition
green_lower = np.array([36, 0,0], dtype = "uint8")
green_upper = np.array([86, 255, 255], dtype = "uint8")
gray_lower = np.array([0, 0, 0], dtype = "uint8")
gray_upper = np.array([255, 65, 160], dtype = "uint8")

# Main cycle
rawFrame = cv2.imread('gas2.png')
# Normalize image brightness
cv2.normalize(rawFrame, rawFrame, 0, 255, cv2.NORM_MINMAX)
# Blur the image to reduce noise
blur = cv2.medianBlur(rawFrame, 5)
# Convert BGR to HSV
hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
# Find color masks
white_mask = cv2.inRange(hsv, gray_lower, gray_upper)
green_mask = cv2.inRange(hsv, green_lower, green_upper)
#full_mask = red2_mask + red3_mask
full_mask = white_mask
#cv2.imshow("debug", full_mask)
# Find counters
contours, hierarchy = cv2.findContours(full_mask, cv2.RETR_LIST,
↪  cv2.CHAIN_APPROX_SIMPLE)
numCon = 0
orient_x = []
orient_y = []
maxlen = 0;
truecont = contours[0];
for icontour in contours:
    if len(icontour) > maxlen:
        maxlen = len(icontour)
        truecont = icontour

ellipse = cv2.fitEllipse(truecont)
cv2.ellipse(rawFrame, ellipse, (0,255,0),2)
hsv = cv2.cvtColor(rawFrame, cv2.COLOR_BGR2HSV)
green_mask = cv2.inRange(hsv, green_lower, green_upper)
contours, hierarchy = cv2.findContours(green_mask, cv2.RETR_LIST,
↪  cv2.CHAIN_APPROX_SIMPLE)
numCon = 0
orient_x = []
orient_y = []
maxlen = 0;
truecont = contours[0];
for icontour in contours:
    if len(icontour) > maxlen:
        maxlen = len(icontour)
        truecont = icontour
cv2.drawContours(rawFrame, truecont, -1, (255,0,0), 2)
cv2.imshow("debug1", rawFrame)


sum_x = 0
sum_y = 0
for i in range(0, len(truecont)):
    sum_x += truecont[i][0][0]
    sum_y += truecont[i][0][1]
avg_x = int(round(sum_x / len(truecont)))
```

```
avg_y = int(round(sum_y / len(truecont)))
print("Contour center location: {} hor, {} ver pixels from the corner".format(avg_x, avg_y))
dxp = avg_x - width / 2
dyp = height/2 - avg_y
print("Contour center location: {} hor, {} ver pixels from the center".format(dxp, dyp))

pitch_rad = uavPitch * math.pi / 180.0
shift_forward = math.tan(pitch_rad) * H
print("Image center is displaced by {:.2f} meters".format(shift_forward))

dxm = dxp * dh
dym = dyp * dv
print("Contour center location: {} hor, {} ver meters from the center".format(dxm, dym))

def get_coords(xp, yp):
    alpha = math.atan(yp / H)
    L = H / math.cos(pitch_rad + alpha)
    l = H / math.cos(pitch_rad)
    dvert = math.sqrt( L**2 + l**2 - 2*L*l*math.cos(alpha))

    beta = math.atan(xp / H)
    Lh = H / math.cos(pitch_rad) / math.cos(beta)
    dhor = math.sqrt( Lh**2 + l**2 - 2*Lh*l*math.cos(beta))
    return (dhor, dvert)

(dhor, dvert) = get_coords(dxm, dym)
print("Object center is dispalced by {:.2f} {:.2f} meters from image center".format(dhor, dvert))
center_x = dhor
center_y = dvert + shift_forward
print("Object center coords: {:.2f} {:.2f}".format(center_x, center_y))

points_x = []
points_y = []
coords_x = []
coords_y = []
print(ellipse)
for i in range(0, len(truecont)):
    dx = truecont[i][0][0] - width / 2
    dy = height/2 - truecont[i][0][1]
    dx  = dx * dh
    dy = dy * dv
    (cx, cy) = get_coords(dx, dy)
    coords_x.append(cx)
    coords_y.append(cy + shift_forward)
    #print("{:.2f} {:.2f}".format(cx - center_x, cy + shift_forward- center_y))
    points_x.append(cx - center_x)
    points_y.append(cy + shift_forward - center_y)

S = 0.5 * (points_y[0]*points_x[-1] - points_y[-1]*points_x[0]);
for i in range(0, len(points_x) - 1):
    S += 0.5 * (points_y[i]*points_x[i+1] - points_y[i+1]*points_x[i])
S = abs(S)
print("AREA: {:.1f}".format(S))
cv2.waitKey()
```

Задача VI.2.4.8.

При решении задач автоматического экологического мониторинга важным фактором является не только текущее состояние окружающей среды, но и прогнозирование его на длительный период времени.

В случае наличия дрейфующего облака вредных веществ, для прогноза его перемещения требуется определять траекторию его движения.

Условие

Разработайте программу на Python, реализующую автоматическое определения траектории движения дрейфующего облака вредных веществ на основе анализа текущих навигационных параметров БПЛА и измерений бортового газоанализатора. В начальный момент времени облако начинает движение из точки происшествия, координаты который известны. Участники могут задавать текущую путевую точку для системы автопилота БПЛА.

Формат входных данных

Координаты точки происшествия; система автоматического управления БПЛА; параметры телеметрии БПЛА в каждый момент времени.

Формат выходных данных

Программа на Python, реализующая автоматический расчёт траектории движения облака вредных веществ.

Критерии оценивания

Оценивается среднее отклонение траектории облака, рассчитанной участниками, от реальной его траектории на интервале времени [0, 120] с.

1. При значении ошибки меньше 20 м за решение задачи начисляется 24 балла.
2. За каждые дополнительные 10 м ошибки снимается 1 балл.
3. Правильный ответ на вопрос по теории прибавляет 1 балл (всего 2 вопроса).

Решение

1. Сформировать управление для достижения облака вредных веществ.
2. Определить координаты точек для полёта по пересекающейся траектории.
3. Определять координаты облака в моменты пересечения траекторий.
4. Записать координаты в файл.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```python
def weighted_mean(values, weights):
    s = 0
    mv = sum(weights)
    for i in range(0, len(values)):
        s += values[i] * weights[i] / mv
    return s

# Поток обработки данных с датчика
def sensor_processing_thread():
    points_x = []
    points_z = []
    last_measurement = 0
    # sendNewWaypoint(500, 500, 100)
    #targets = [(180, 200, 80), (50, 220, 80), (300, 600, 80), (0, 500, 80), (-80, 700, 80), (-300, 550, 80), (-400, 700, 80), (-100, 0, 80)]
    targets = [(200, 220, 80), (240, 240, 80), (60, 250, 80), (260, 650, 80), (-50, 550, 80), (-180, 730, 80), (-300, 550, 80), (-400, 700, 80), (-100, 0, 80)]
    #cpt = targets
    count = 0
    dist = 0
    sendNewWaypoint(targets[0][0], targets[0][1], targets[0][2])
    traj = []
    rx = []
    ry = []
    rw = []
    inside = False
    c = 0
    step = 600
    working = False
    reach = False
    dira = 10
    with open('test\\uav.csv', 'w', newline='\n') as fd:
        writer = csv.writer(fd, delimiter=",")
        writer.writerow([0, 0, 0])

    # Основной цикл программы
    while running:
        tele = last_telemetry
        dist = sqrt((targets[count][0] - tele.x)**2 + (targets[count][1] - tele.z)**2)
        if dist <= 15:
            count = (count + 1) % len(targets)
            if (count >= len(targets)):
                count = 0
            sendNewWaypoint(targets[count][0], targets[count][1], targets[count][2])

        #if tele.air > 0.85:
        #    print(f"c {tele.x:.2f} {tele.z:.2f}")
        if tele.air > 0:
            inside = True
        elif tele.air == 0 and last_measurement != 0 and len(rw) > 0:
            inside = False
            traj.append((weighted_mean(rx, rw), weighted_mean(ry, rw)))
            with open('test\\uav.csv', 'a', newline='\n') as fd:
                writer = csv.writer(fd, delimiter=",")
                writer.writerow([tele.time, weighted_mean(rx, rw), weighted_mean(ry, rw)])
            print(f"val: {max(rw)}")
```

```
            rx = []
            ry = []
            rw = []

            print(traj[-1])
            print("---")

        if inside and tele.air > 0.7:
            rx.append(tele.x )
            ry.append(tele.z)
            rw.append(tele.air)

        last_measurement = tele.air
        time.sleep(0.05)
```