Командный тур

5.1. Задачи

Система оценки

Задача 1:

- Задание выполнено полностью верно = 5 баллов
- Задание выполнено, но есть недочеты = 1 . . . 4 балла
- Задание выполнено неверно = 0 баллов
- Ответ на дополнительные вопросы = +2 балла

Задача 2:

- Задание выполнено полностью верно = 7 баллов
- Задание выполнено, но есть недочеты = 1 . . . 6 баллов
- Задание выполнено неверно = 0 баллов
- Ответ на дополнительные вопросы = +2 балла

Задача 3:

- Задание выполнено полностью верно = 4 балла
- Задание выполнено, но есть недочеты = 1 . . . 3 балла
- Задание выполнено неверно = 0 баллов
- Ответ на дополнительные вопросы = +2 балла

Задача 4:

- Задание выполнено полностью верно = 6 баллов
- Задание выполнено, но есть недочеты = 1 . . . 5 баллов
- Задание выполнено неверно = 0 баллов
- Ответ на дополнительные вопросы = +2 балла

Задача 5:

- Задание выполнено полностью верно = 10 баллов
- Задание выполнено, но есть недочеты = 1 . . . 9 баллов
- Задание выполнено неверно = 0 баллов

Задача 6:

- Задание выполнено полностью верно = 7 баллов
- Задание выполнено, но есть недочеты = 1 . . . 6 баллов

Задание	Задачи					
	Задача 1	Задача 2	Задача 3	Задача 4	Задача 5	Задача 6
Максимальный балл по каждой задаче	7	9	6	8	10	7

Задача 5.1.1. (7 баллов)

Напишите программу управления ШИМ сигналом через потенциометр. Длительность импульса ШИМ сигнала должна изменяться от 1000 до 2000 мкс в зависимости от напряжения на выходе потенциометра (1000 мкс для минимального выходного напряжения и 2000 мкс для максимального). Соберите схему и при помощи осциллографа снимите статическую характеристику напряжение-длительность импульса ШИМ сигнала

Решение

1. Подключить потенциометр к контроллеру по следующей схеме:

Рис. 1 Схема подключения потенциометра.

2. Определить реальное максимальное U_max и минимальное U_min выходное напряжение потенциометра. Программа читает отсчеты АЦП на ножке A0 и определяет максимальное и минимальное значение. Во время этого ручку потенциометра требуется вращать из одного крайнего положения в другое.
3. Рассчитать коэффициенты линейного преобразования $PWM = kU_{\text{вх}} + b$:

$$k = \frac{1000}{U_{\max} - U_{\min}}$$

$$b = 1000 \left(1 - \frac{U_{\min}}{U_{\max} - U_{\min}}\right),$$

где PWM – длительность импульса выходного ШИМ сигнала, $U_{вх}$ – входное напряжение на ножку A0. Число 1000 – минимальное значение PWM

4. В основном цикле программы провести чтение входного напряжение $U_{вх}$ на ножку A0, рассчитать значение $PWM = kU_{вх} + b$, вывести ШИМ сигнал на цифровую ножку 2 с длительностью импульса PWM (мкс).

Пример программы-решения

Ниже представлено решение на языке C++

```
// Напишите программу управления ШИМ сигналом через потенциометр.
// Длительность импульса ШИМ сигнала должна изменяться от 1000 до 2000 мкс
// в зависимости от напряжения на выходе потенциометра (1000 мкс для
// минимального выходного напряжения и 2000 мкс для максимального).
// Соберите схему и при помощи осциллографа снимите статическую характеристику
// напряжение-длительность импульса ШИМ сигнала.

#include <math.h>
#include <Servo.h>

// Аналоговая ножка для приема сигнала с потенциометра
const uint8_t VRPIN = A0;

// Отсчеты АЦП (0 - 1023)
uint16_t adcCounts = 0;

// Цифровая ШИМ ножка
const uint8_t SRVPIN = 2;

// Цифровой ШИМ выход
Servo CTRL;

// Выходное значение (1000 - 2000 мкс)
uint16_t pwmCounts = 0;

// Коэффициенты линейной зависимости пересчета отсчеты АЦП - ШИМ
float k = 0;
float b = 0;

// Максимальное и минимальное значение напряжений с потенциометра
// Установлены такие значения, что они гарантированно перезапишутся при
// их экспериментальном определении
uint16_t adcCounts_min = 1024;
uint16_t adcCounts_max = 0;

void setup()
{
  // Настройка портов ввода-вывода
  Serial.begin(115200); while(!Serial) { ; }
  pinMode(VRPIN,INPUT);
  CTRL.attach(SRVPIN);

  // Определение минимального и максимального значений
  Serial.print("Rotate potentiometer/n");

  uint32_t startTime = millis();
  while (millis() - startTime < 5000)
  {
```

```
    adcCounts = analogRead(VRPIN);
    if (adcCounts < adcCounts_min) adcCounts_min = adcCounts;
    if (adcCounts > adcCounts_max) adcCounts_max = adcCounts;
  }

  Serial.print("adcCounts_min = " + String(adcCounts_min) + "\n");
  Serial.print("adcCounts_max = " + String(adcCounts_max) + "\n");

  // Расчет коэффициентов преобразования
  k = 1000.0 / float(adcCounts_max - adcCounts_min);
  b = 1000.0 * (1.0 - float(adcCounts_min) / float(adcCounts_max - adcCounts_min));
}

void loop()
{
  // Чтение входного напряжения
  adcCounts = analogRead(VRPIN);

  // Расчет выходного ШИМ
  pwmCounts = round(k* float(adcCounts) + b);

  // Ограничение ШИМ на случай погрешностей округления
  if (pwmCounts < 1000) pwmCounts = 1000;
  if (pwmCounts > 2000) pwmCounts = 2000;

  // Вывод ШИМ
  CTRL.writeMicroseconds(pwmCounts);

  // Для отладки
  Serial.print("adcCounts = " + String(adcCounts) + "\n");
  Serial.print("pwmCounts = " + String(pwmCounts) + "\n");

  // 20 Гц
  delay(50);
}
```

Задача 5.1.2. (9 баллов)

Напишите программу определения угла крена комплексируя измерения акселерометра и гироскопа. Проведите предварительную ручную калибровку акселерометра и гироскопа. Калибровку акселерометра проведите с учетом сдвига нуля и масштабного коэффициента, при калибровке гироскопа учтите только сдвиг нуля. Выведите на один график оценку угла крена по акселерометру и комплексированную. Подберите коэффициент фильтрации так. чтобы в измерения не вносилось запаздывание, но точность улучшилась.

Решение

Этап 1. Программа калибровки датчиков

Модели измерений датчиков имеют вид:

$$\omega_i = \omega_i^{\text{изм}} - b_i^{\text{гир}} - \xi_i,$$

$$n_i = k_i^{\text{акс}}(n_i^{\text{изм}} - b_i^{\text{акс}}) - \eta_i,$$

где ω_i – истинное значение угловой скорости вращения вокруг оси $i = x, y, z$, $\omega_i^{\text{изм}}$ – измеренное значение ω_i, $b_i^{\text{гир}}$ – сдвиг нуля гироскопа, n_i – истинное значение кажущегося ускорения по оси i, $n_i^{\text{изм}}$ – измеренное значение n_i, $b_i^{\text{акс}}$ – сдвиг нуля акселерометра, $k_i^{\text{акс}}$ – масштабный коэффициент акселерометра, ξ_i, η_i – случайные погрешности гироскопа и акселерометра соответственно.

1. Калибровка гироскопа. Оставить датчик неподвижно – истинная угловая скорость в этом случае принимается за 0. Для каждой оси найти значение масштабного коэффициента $b_i^{\text{гир}}$ как среднее N замеров $\omega_{i,k}^{\text{изм}}$ ($N > 100, k$ – номер замера):

$$b_i^{\text{гир}} = \frac{1}{N} \sum_{k=1}^{N} \omega_{i,k}^{\text{изм}}$$

2. Калибровка акселерометра. Для каждой оси акселерометра вычислить следующие два средних значения:

$$\mu_j = \frac{1}{N} \sum_{k=1}^{N} n_{i,k}^{\text{изм}}, j = 1, 2,$$

причем значение μ_1 считается, когда соответствующая ось акселерометра направлена вверх, а значение μ_2 когда ось направлена вниз. За истинные значения μ_j принимаются соответственно $+g$ и $-g$, где $g \cong 9.81$ м/с^2. Используя модель измерений акселерометра и пренебрегая случайной погрешностью, получим следующую систему двух уравнений для каждой оси акселерометра:

$$g = k_i^{\text{акс}}(\mu_1 - b_i^{\text{акс}}),$$

$$-g = k_i^{\text{акс}}(\mu_2 - b_i^{\text{акс}}).$$

Решив данную систему относительно $k_i^{\text{акс}}$ и $b_i^{\text{акс}}$ получим следующий результат:

$$k_i^{\text{акс}} = \frac{2g}{\mu_2 - \mu_1},$$

$$b_i^{\text{акс}} = \frac{\mu_1 + \mu_2}{2}.$$

В программе выбор оси для снятия замеров осуществляется вводом соответствующего символа с клавиатуры.

Этап 2. Программа расчета угла крена

1. Задать найденные в прошлом этапе калибровочные коэффициенты в класс, осуществляющий получение данных с датчика MPU9250 с помощью функций setAccelCal и setGyroBias_rads.
2. Найти угловую скорость крена на текущем шаге k : $w_k = -\omega_x$.
3. Найти измерение угла крена по акселерометру: $z_k = atan2(n_{y'} - n_z)$.
4. Провести комплексирование измерений по формуле:

$$\hat{x}_k = Kz_k + (1 - K)(x_k + w_k \Delta t),$$

где $\hat{x}_k$- оценка значения угла крена на текущем шаге, Δt – шаг дискретизации, K – коэффициент усиления. Подбор коэффициента усиления осуществляется

следующим образом. Коэффициент увеличивают (при подавляется шум измерений) пока алгоритм комплексирования не начнет вносить запаздывания в систему измерений на существенных частотах, т.е. запаздывание не заметно при обычных маневрах самолета. Например, для $\Delta t = 20$ мс хорошие результаты дает значение коэффициента $K = 0.07 - 0.13$.

5. Вывести в последовательный порт значения z_k и $\hat{x}_k$.

Код программы калибровки датчиков

```
// Программа ручной калибровки датчика MPU9250
#include "MPU9250.h"
#include "common.h"

//============================================================================
// Переменные

// Датчик MPU9250, адрес 0x68, шина I2C
MPU9250 IMU(Wire,0x68);
int status;

//----------------------------------------------------------------------------
// измерения
float nx, ny, nz;         // кажущиеся ускорения [м/с2]
float wx, wy, wz;         // угловые скорости [рад/с]

// буфер замеров
const int N = 250;
float mes[N];

//----------------------------------------------------------------------------
// параметры калибровки

// гироскоп
float wx_b, wy_b, wz_b; // сдвиг нуля гироскопа

// акселерометр
float muxp, muxm;         // средние значения мин и макс измерений
float muyp, muym;
float muzp, muzm;
float kx, ky, kz;         // масштабный коэффициент акселерометра
float bx, by, bz;         // сдвиг нуля акселерометра

// буфер для чтения входной строки
uint8_t symbol;

// функции настройки и чтения
extern void setupImu();
extern void readImu();

//============================================================================
// Основная функция: настройка программы
void setup() {
  // открытие последовательного порта
  Serial.begin(115200); while(!Serial) { ; }

  // настройка датчика
  setupImu();

  // калибровка гироскопа: сдвиг нуля рассчитывается как среднее N замеров по каждой оси
```

```
51    for (int k = 0; k < N; k++)
52    {
53      readImu();
54      shiftArray(mes, N, wx);
55      delay(20);
56    }
57    wx_b = mean(mes, N);
58
59    for (int k = 0; k < N; k++)
60    {
61      readImu();
62      shiftArray(mes, N, wy);
63      delay(20);
64    }
65    wy_b = mean(mes, N);
66
67    for (int k = 0; k < N; k++)
68    {
69      readImu();
70      shiftArray(mes, N, wz);
71      delay(20);
72    }
73    wz_b = mean(mes, N);
74
75    // вывод сдвигов нуля гироскопа в порт
76    Serial.print("gyro bias X = " + String(wx_b,6) + "\n");
77    Serial.print("gyro bias Y = " + String(wy_b,6) + "\n");
78    Serial.print("gyro bias Z = " + String(wz_b,6) + "\n");
79
80  }
81
82  //=============================================================================
83  // основная функция: цикл
84  void loop() {
85    // Калибровка акселерометра
86    if (Serial.available() > 0)
87    {
88      // считать символ
89      symbol = Serial.read();
90
91      // очистить буфер
92      while (Serial.available() > 0) Serial.read();
93
94      // определить калибруемую ось
95      // OX ----------------------------------------
96      if (symbol == 'q')
97      {
98        for (int k = 0; k < N; k++)
99        {
100         readImu();
101         shiftArray(mes, N, nx);
102         delay(20);
103       }
104       muxp = mean(mes, N);
105       Serial.print("muxp = " + String(muxp) + "\n");
106     }
107
108     if (symbol == 'w')
109     {
110       for (int k = 0; k < N; k++)
```

```
111         {
112           readImu();
113           shiftArray(mes, N, nx);
114           delay(20);
115         }
116         muxm = mean(mes, N);
117         Serial.print("muxm = " + String(muxm) + "\n");
118       }
119
120       // OY ---------------------------------------
121       if (symbol == 'e')
122       {
123         for (int k = 0; k < N; k++)
124         {
125           readImu();
126           shiftArray(mes, N, ny);
127           delay(20);
128         }
129         muyp = mean(mes, N);
130         Serial.print("muyp = " + String(muyp) + "\n");
131       }
132
133      if (symbol == 'r')
134       {
135         for (int k = 0; k < N; k++)
136         {
137           readImu();
138           shiftArray(mes, N, ny);
139           delay(20);
140         }
141         muym = mean(mes, N);
142         Serial.print("muym = " + String(muym) + "\n");
143       }
144
145       // OZ ---------------------------------------
146       if (symbol == 't')
147       {
148         for (int k = 0; k < N; k++)
149         {
150           readImu();
151           shiftArray(mes, N, nz);
152           delay(20);
153         }
154         muzp = mean(mes, N);
155         Serial.print("muzp = " + String(muzp) + "\n");
156       }
157
158       if (symbol == 'y')
159       {
160         for (int k = 0; k < N; k++)
161         {
162           readImu();
163           shiftArray(mes, N, nz);
164           delay(20);
165         }
166         muzm = mean(mes, N);
167         Serial.print("muzm = " + String(muzm) + "\n");
168       }
169
170       // finish
```

```
171     if (symbol == 'f')
172     {
173                 // расчет калибровочных параметров акселерометра
174       kx = 19.614/(muxp - muxm); bx = (muxp + muxm)/2.0;
175       ky = 19.614/(muyp - muym); by = (muyp + muym)/2.0;
176       kz = 19.614/(muzp - muzm); bz = (muzp + muzm)/2.0;
177 
178       Serial.print("acc bias X = " + String(bx,6) + "\t"
179         + "acc scale X = "+ String(kx,6) + "\n");
180       Serial.print("acc bias Y = " + String(by,6) + "\t"
181         + "acc scale Y = "+ String(ky,6) + "\n");
182       Serial.print("acc bias Z = " + String(bz,6) + "\t"
183         + "acc scale Z = "+ String(kz,6) + "\n");
184 
185       IMU.setAccelCalX(bx, kx);
186       IMU.setAccelCalY(by, ky);
187       IMU.setAccelCalZ(bz, kz);
188 
189       IMU.setGyroBiasX_rads(wx_b);
190       IMU.setGyroBiasY_rads(wy_b);
191       IMU.setGyroBiasZ_rads(wz_b);
192 
193           // переход в нормальный режим работы
194       while (1)
195       {
196         // read the sensor
197         IMU.readSensor();
198 
199         // display the data
200         Serial.print(IMU.getAccelX_mss(),6);  Serial.print("\t");
201         Serial.print(IMU.getAccelY_mss(),6);  Serial.print("\t");
202         Serial.print(IMU.getAccelZ_mss(),6);  Serial.print("\t");
203         Serial.print(IMU.getGyroX_rads(),6);  Serial.print("\t");
204         Serial.print(IMU.getGyroY_rads(),6);  Serial.print("\t");
205         Serial.print(IMU.getGyroZ_rads(),6);  Serial.print("\n");
206         delay(20);
207       }
208     }
209 
210   }
211 
212 }
213 
214 //================================================================================
215 // настройка датчика
216 void setupImu()
217 {
218   status = IMU.begin();
219   if (status < 0) {
220     Serial.println("IMU initialization unsuccessful");
221     Serial.println("Check IMU wiring or try cycling power");
222     Serial.print("Status: ");
223     Serial.println(status);
224     while(1) {}
225   }
226 
227   IMU.setAccelRange(MPU9250::ACCEL_RANGE_2G);
228   IMU.setGyroRange(MPU9250::GYRO_RANGE_250DPS);
229   IMU.setDlpfBandwidth(MPU9250::DLPF_BANDWIDTH_20HZ);
230 
```

```
  // 50 Hz update rate
  IMU.setSrd(19);
}

//===========================================================================
// чтение датчика
void readImu()
{
  IMU.readSensor();

  nx = IMU.getAccelX_mss();
  ny = IMU.getAccelY_mss();
  nz = IMU.getAccelZ_mss();

  wx = IMU.getGyroX_rads();
  wy = IMU.getGyroY_rads();
  wz = IMU.getGyroZ_rads();

}
```

Код программы оценки угла крена

```
// программа определения угла крена на основе комплексирования измерений
// акселерометра и гироскопа с использованием фильтра Калмана
#include "MPU9250.h"

//===========================================================================
// Переменные

// Датчик MPU9250, адрес 0x68, шина I2C
MPU9250 IMU(Wire,0x68);
int status;

// кажущиеся ускорения [м/с2]
float nx, ny, nz;

// угловые скорости [рад/с]
float wx, wy, wz;

// параметры калибровки (задаются вручную по результатам предыдущей программы)
float gyroBiasX = 0.000174;
float gyroBiasY = 0.000219;
float gyroBiasZ = 0.000143;

float accelBiasX = 0.113216;
float accelBiasY = 0.180062;
float accelBiasZ = -0.599224;

float accelScaleFactorX = 1.000751;
float accelScaleFactorY = 0.998263;
float accelScaleFactorZ = 0.991242;

// функции настройки и чтения
extern void setupImu();
extern void readImu();

// параметры оценивателя
float K = 0.1;                  // коэффициент усиления
float xhatk = 0;          // оценка угла на текущем шаге
float xhatk_1 = 0;          // оценка угла на предыдущем шаге
```

```
39 float zk = 0.0;                 // измерение угла по акселерометру
40 float wk = 0.0;                 // угловая скорость на текущем шаге
41 float wk_1 = 0.0;          // угловая скорость на предыдущем шаге
42 unsigned long int t;          // время последнего вызова фильтра
43
44 //============================================================================
45 // Основная функция: настройка программы
46 void setup()
47 {
48   // открытие последовательного порта
49   Serial.begin(115200); while(!Serial) { ; }
50
51   // настройка датчика
52   setupImu();
53 }
54
55 //============================================================================
56 // основная функция: цикл
57 void loop()
58 {
59         // чтение датчика
60         readImu();
61
62         // процедура определения угла крена
63         wk = -wx;          // угловая скорость вокруг оси Ox связанной (!)
64             //системы координат
65         zk = atan2(ny,-nz);          // угол крена по измерениям акселерометра
66         // вычисление оценки: средневзвешенное между измерением и прогнозом
67         xhatk = K*zk + (1.0 - K)*(xhatk_1 + wk_1*float(micros()-t)*1.0e-6);
68
69         // сдвиг замеров
70         t = micros();
71         wk_1 = wk;
72         xhatk_1 = xhatk;
73
74         // вывод данных
75         Serial.println(String(zk*57.3,6) +"\t"+ String(xhatk*57.3,6) +"\t"
76     + String(wk*57.3,6));
77
78         // 50 Hz
79         delay(20);
80 }
81
82 //============================================================================
83 // настройка датчика
84 void setupImu()
85 {
86   status = IMU.begin();
87   if (status < 0) {
88     Serial.println("IMU initialization unsuccessful");
89     Serial.println("Check IMU wiring or try cycling power");
90     Serial.print("Status: ");
91     Serial.println(status);
92     while(1) {}
93   }
94
95   IMU.setAccelRange(MPU9250::ACCEL_RANGE_2G);
96   IMU.setGyroRange(MPU9250::GYRO_RANGE_250DPS);
97   IMU.setDlpfBandwidth(MPU9250::DLPF_BANDWIDTH_184HZ);
98
```

```
99    // 50 Hz update rate
100   IMU.setSrd(19);
101
102   IMU.setAccelCalX(accelBiasX, accelScaleFactorX);
103   IMU.setAccelCalY(accelBiasY, accelScaleFactorY);
104   IMU.setAccelCalZ(accelBiasZ, accelScaleFactorZ);
105
106   IMU.setGyroBiasX_rads(gyroBiasX);
107   IMU.setGyroBiasY_rads(gyroBiasY);
108   IMU.setGyroBiasZ_rads(gyroBiasZ);
109 }
110
111 //============================================================================
112 // чтение датчика
113 void readImu()
114 {
115   IMU.readSensor();
116
117   nx = IMU.getAccelX_mss();
118   ny = IMU.getAccelY_mss();
119   nz = IMU.getAccelZ_mss();
120
121   wx = IMU.getGyroX_rads();
122   wy = IMU.getGyroY_rads();
123   wz = IMU.getGyroZ_rads();
124 }
```

Задача 5.1.3. (6 баллов)

Напишите программу взаимодействия с органами управления БПЛА. Требуется запустить двигатель на 3 сек. и выключить его, отклонить руль высоты в крайнее верхнее положение на 3 сек., затем в крайнее нижнее на 3 сек. и вернуть в нейтральное. Последнее повторить для руля направления и элеронов. Для элеронов первое отклонение должно обеспечить положительный крен. Для руля направления – положительное изменение курса.

Решение

1. Конфигурировать цифровые ножки контроллера согласно следующей таблице.

Таблица 5.1: Таблица соответствий каналов управления и цифровых выводов контроллера.

Канал управления	Цифровой вывод
Элероны	2
Руль высоты	3
Двигатель	5
Руль направления	6

2. Запустить двигатель на 3 сек. Для этого требуется экспериментально определить значение ШИМ на котором двигатель запускается (1200). С целью уменьшения нагрузки на двигатель, ШИМ увеличивается линейно со временем в течение 3х секунд с стартового значения до максимально разрешенного

(1500), после чего ШИМ устанавливается в значение 1000, что соответствует выключению двигателя.

3. Элероны и рули высоты и направления управляются однотипно. Крайние положения соответствуют значениям ШИМ 1000 и 2000. Значения ШИМ для положительного отклонения управляющих поверхностей определяются экспериментально. В данном случае положительные отклонения соответствуют значению 2000.
4. Для перевода поверхностей в нейтральное положение значению ШИМ присваивается 1500.

Пример программы-решения

Ниже представлено решение на языке C++

```
#include <Servo.h>

// Стандартные значения ШИМ
const uint16_t PWM_MIN = 1000;
const uint16_t PWM_MAX = 2000;
const uint16_t PWM_AVG = 1500;

// ШИМ запуска двигателя
const uint16_t PWM_ENG_START = 1200;

// Максимальный ШИМ для двигателя (ШИМ на максимально разрешенной тяге)
const uint16_t PWM_ENG_MAX = 1500;

// Коэффициент нарастания ШИМ (тяга) [ШИМ/сек]
const float k = 150;

// Стандартное время работы (мс)
const uint16_t OPTIME = 3000;

// ШИМ каналы
Servo AIL;  // элероны
Servo ELE;  // руль высоты
Servo THR;  // тяга двигателя
Servo RUD;  // руль направления


void setup()
{
  //============================================================================
  // Порт для оталадки
  Serial.begin(115200); while(!Serial) { ; }

  // подключение каналов управления
  AIL.attach(2);  AIL.write(PWM_AVG);
  ELE.attach(3);  ELE.write(PWM_AVG);
  THR.attach(5);  THR.write(PWM_MIN);
  RUD.attach(6);  RUD.write(PWM_AVG);


  //============================================================================
  // 1. Двигатель

```

```
43    // Подкинуть ШИМ до стартового
44    THR.writeMicroseconds(PWM_ENG_START);
45
46    // Во избежание несчастных случаев на производстве ШИМ изменять линейно
47    uint32_t tstart = millis();
48    uint16_t pwm = 0;
49
50    pwm = PWM_ENG_START + k*float(millis()  - tstart)/1000.0;
51    while (millis()  - tstart < OPTIME)
52    {
53      // Ограничение ШИМ
54      if (pwm < PWM_ENG_MAX)
55      {
56        pwm = PWM_ENG_MAX;
57      }
58
59      THR.writeMicroseconds(pwm);
60
61      // 20 Hz
62      delay(50);
63    }
64    THR.writeMicroseconds(PWM_MIN);
65    delay(1000); //задержка между выключением двигателя и подачей сигналов
66    // на органы управления
67    //============================================================================
68    // 2. Органы управления
69
70    //----------------------------------------------------------------------------
71    // 2.1. Руль высоты
72    Serial.print("ELE UP\n");
73    ELE.writeMicroseconds(PWM_MAX);
74    delay(OPTIME);
75
76    Serial.print("ELE DOWN\n");
77    ELE.writeMicroseconds(PWM_MIN);
78    delay(OPTIME);
79
80    Serial.print("ELE CENTER\n\n");
81    ELE.writeMicroseconds(PWM_AVG);
82    delay(OPTIME);
83
84    //----------------------------------------------------------------------------
85    // 2.2. Руль направления
86    Serial.print("RUD UP\n");
87    RUD.writeMicroseconds(PWM_MAX);
88    delay(OPTIME);
89
90    Serial.print("RUD DOWN\n");
91    RUD.writeMicroseconds(PWM_MIN);
92    delay(OPTIME);
93
94    Serial.print("RUD CENTER\n\n");
95    RUD.writeMicroseconds(PWM_AVG);
96    delay(OPTIME);
97
98    //----------------------------------------------------------------------------
99    // 2.3. Элероны
100   Serial.print("AIL UP\n");
101   AIL.writeMicroseconds(PWM_MAX);
102   delay(OPTIME);
```

```
103
104   Serial.print("AIL DOWN\n");
105   AIL.writeMicroseconds(PWM_MIN);
106   delay(OPTIME);
107
108   Serial.print("AIL CENTER\n\n");
109   AIL.writeMicroseconds(PWM_AVG);
110   delay(OPTIME);
111
112 }
113
114 void loop()
115 {
116   // пустой цикл
117 }
```

Задача 5.1.4. (8 баллов)

Спроектируйте систему управления углом курса БПЛА через крен на симуляторе полета БПЛА в системе MATLAB. Система управления должна удовлетворять требованиям:

- время переходного процесса в линейной зоне – 10 сек;
- перанный угол крена не более 30 градусов по модулю;
- разерегулирование не более 5%;
- задворот должен проходить по кратчайшему пути.

Решение

Управление по курсу осуществляется через изменение угла крена БПЛА. Другими словами, входом системы управления является рассогласование $\psi - \psi^{\text{зад}}$, где ψ – текущий угол курса, $\psi^{\text{зад}}$ – требуемый угол курса, а выходной величиной – заданный угол крена $\gamma^{\text{зад}}$.

Подходящим контроллером является пропорциональный регулятор (П-регулятор):

$$u = K_\psi \epsilon,$$

где ϵ – сигнал ошибки, u – управляющий сигнал, K_ψ – коэффициент усиления.

Оба угла ψ и $\psi^{\text{зад}}$ изменяются в пределах $[-180°, +180°]$. Для обеспечения кратчайшего разворота сигнал ошибки ϵ рассчитывается как

$$\epsilon = atan2(sin(\psi - \psi^{\text{зад}}), cos(\psi - \psi^{\text{зад}}))$$

.

Чтобы заданный угол крена не превышал по модулю 30 градусов, управляющий сигнал регулятора ограничивается следующим образом:

$$\gamma^{\text{зад}} = \begin{cases} u, |u| \leqslant 30° \\ 30°, u > 30° \\ -30°, u < -30° \end{cases}$$

Коэффициент K_ψ подбирается таком малом сигнале рассогласования, что $|\gamma^{\text{зад}} < 30°$. Увеличение коэффициента ускоряет переходный процесс, однако, при больших значениях K_ψ регулятор курса может работать быстрее, чем система стабилизации крена, что приведет к перерегулированию и колебательности. Удовлетворительное качество процесса регулирования достигается при $K_\psi = 8.14$.

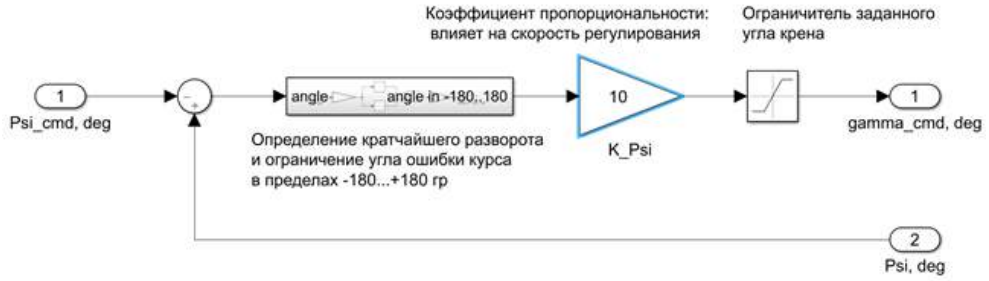

Рис. 2 Структурная схема системы управления углом курса.

Задача 5.1.5. (10 баллов)

Напишите программу полета БПЛА на заданную точку. Полет на заданную точку осуществить при помощи регулятора курса, разработанного в задаче 4. Решение задачи проверить в ходе летных испытаний.

Решение

1. Рассчитать курс на требуемую точку. используя измерения текущего местоположения.
2. Рассчитать расстояние до требуемой точки, используя измерения текущего местоположения.
3. Если расстояние до заданной точки меньше заданного (5 м), подать сигнал на цифровую ножку для включения камеры.
4. Провести расчет заданного угла курса для стабилизации курса на точку аналогично Задаче 4. Полученный ответ перевести из градусов в отсчеты ШИМ и ограничить.
5. Вывести полученное значение ШИМ сигнала в канал элеронов. Параметры системы управления:
   - Коэффициент пропорциональности регулятора: 4;
   - Ограничения ШИМ в канале элеронов: 1350... 1650;
   - Коэффициент пересчета градусы – ШИМ: 2.

Пример программы-решения

Ниже представлено решение на языке C++

```
1 //=========================================================================
2 // Подкоючение библиотек
```

```
3
4  //--------------------------------------------------------------------------------
5  // Стандартные
6  #include <SPI.h>
7  #include <SD.h>
8  #include <Servo.h>
9  #include <math.h>
10
11 //--------------------------------------------------------------------------------
12 // Пользовательские
13 #include "common.h"
14 #include "TinyGPSpp.h"
15
16 //================================================================================
17 // Определения
18
19 // Если определено, то система ждет сигнала с ГНСС, проверяет его валидность и
20 // рассчитывает стартовую точку. Для тестов в помещении можно закомментировать
21 #define GNSS_ENABLED
22
23 //================================================================================
24 // Периферийные устройства
25
26 //--------------------------------------------------------------------------------
27 // ГНСС (NMEA)
28
29 // основной класс (строка $GPGGA)
30 TinyGPSPlus gps;
31
32 // дополнительные классы
33 // индикатор захвата позиции
34 TinyGPSCustom TGPSC_positionFixIndicator(gps, "GPGGA", 6);
35
36 // поправка по высоте [м]
37 TinyGPSCustom TGPSC_geoidSeparation(gps, "GPGGA", 11);
38
39 // угол пути [градусы]
40 TinyGPSCustom TGPSC_heading_deg(gps, "GPVTG", 1);
41
42 // горизонтальная скорость [км/ч]
43 TinyGPSCustom TGPSC_horizontalVelocity_kh(gps, "GPVTG", 7);
44
45 // скорость последовательного порта
46 const uint32_t GPS_BAUD_RATE = 38400;
47
48 // Измерения
49 const int N_GNSS = 1;     // число замеров для фильтрации
50 const int N_GNSS_0 = 20;  // число точек для вычисления начальной позиции
51
52 // время последнего вызова чтения датчиков [мс]
53 unsigned long int TLC_MS_GNSS = 0;
54 // мгновенные замеры
55 uint8_t hours, minutes, seconds;  // часы, минуты, секунды
56 int nSats;                // число спутников
57 double hdop;              // горизонтальный геометрический фактор
58 int pfi;                // индикатор захвата позиции
59
60 // массивы замеров
61 double latitudes[N_GNSS];          // широта [градусы]
62 double longitudes[N_GNSS];         // долгота [градусы]
```

```
63  double altitudesMsl[N_GNSS];      // высота над уровнем моря [м]
64  double headings[N_GNSS];         // угол пути [рад]
65  double horizontalVelocities[N_GNSS];  // модуль горизонтальной скорости [м/с]
66
67  // осредненные оценки
68  double initialLatitude = 0.0;    // широта точки старта [градусы]
69  double initialLongitude = 0.0;     // долгота точки старта [градусы]
70  double latitude = 0.0;         // широта [градусы]
71  double longitude = 0.0;        // долгота [градусы]
72  double altitudeMsl = 0.0;      // высота над уровнем моря [м]
73  double heading = 0.0;        // угол пути [рад]
74  double horizontalVelocity = 0.0;  // модуль горизонтальной скорости [м/с]
75
76  // рассчитываеме параметры
77  double velocityNorth = 0.0; // северная проекция скорости [м/с]
78  double velocityEast = 0.0;  // восточная проекция скорости [м/с]
79
80  const int N_SATS_MIN = 7;  // минимальное число спутников
81
82  // функция настройки
83  extern void setupMt3333();
84
85  // функция чтения
86  extern void readMt3333();
87
88  //-------------------------------------------------------------------------
89  // SD карта
90  File sdLog;                  // файл записи данных
91  String sdLogData;               // строка данных
92  unsigned long int T_MS_AUTOSAVE = 30000;  // период автосохранения
93  unsigned long int TLC_MS_AUTOSAVE = 0;    // время последнего автосохранения
94  unsigned long int T_MS_SD = 50;       // период записи
95  unsigned long int TLC_MS_SD = 0;        // время последней записи
96
97  extern void sdWrite();  // функция записи
98
99  //=========================================================================
100 // Система управления
101 int pwmAil = 1500;    // значение ШИМ управления
102 Servo AIL;        // ШИМ канал элероны расположены на ножке 2
103 int D_PWM_MAX = 300;  // максимальное изменение ШИМ относительно 1500
104 double eps = 0.0;   // сигнал рассогласования
105
106 // флаг режима полета ручка\автопилот
107 bool mode = false;
108 const int MODE_PIN = 35;
109
110
111 //=========================================================================
112 /***************************** ПОЛЕТ ПО КУРСУ ****************************/
113 float presetCoordinates[2] = {56.097714, 35.878395};        // координаты метки
114 #define SETPOINT_RADIUS 5        // Окрестность точки, в которой провести съемку
115 #define HEADING_P       4        // Коэффициент П регулятора
116 /*************************************************************************/
117
118
119 //=========================================================================
120 // Функция основной программы: инициализация
121 void setup()
122 {
```

```
123   // на всякий случай :P
124   delay(5000);
125   Serial2.print("START\n");
126 
127   //-------------------------------------------------------------------
128   // Инициализация последовательного порта
129   Serial.begin(115200); while (!Serial) {
130     ;  // порт отладки
131   }
132   Serial2.begin(19200); while (!Serial2) {
133     ;  // порт телеметрии
134   }
135   delay(5000);
136   //-------------------------------------------------------------------
137   // Инициализация ГНСС
138   setupMt3333();
139   //-------------------------------------------------------------------
140   // Инициализация SD карты
141   if (SD.begin(SD_CHIP_SELECT_PIN)) Serial2.print("SD RDY\n");
142   else Serial2.print("SD NOT FOUND\n");
143   sdLog = SD.open("log.txt", FILE_WRITE);
144 
145   //-------------------------------------------------------------------
146   // подключение каналов управления
147   AIL.attach(2);  // Прикрепляем ногу
148   AIL.write(1500); // Задаем скорость.
149 
150 
151 
152   pinMode(MODE_PIN, INPUT);
153   pinMode(4, OUTPUT);//ПИН КАМЕРЫ
154 }
155 
156 //=====================================================================
157 // Функция основной программы: цикл
158 void loop()
159 {
160         //-------------------------------------------------------------------
161         // Получение данных с ГНСС приемника
162         readMt3333(); // Тут вы получаете heading, latitude, longitude с GPS
163 
164         /*********************************************************************/
165         /**                          Начало разработки                      **/
166         /*********************************************************************/
167         // Курс на метку
168         double courseToPreset = TinyGPSPlus::courseTo(latitude, longitude,
169                                 presetCoordinates[0], presetCoordinates[1]);
170 
171         // Расстояние до метки
172         unsigned long distanceToPreset =
173                 (unsigned long) TinyGPSPlus::distanceBetween(latitude, longitude,
174                 presetCoordinates[0], presetCoordinates[1]);
175 
176         //Включение камеры в заданном радиусе до точки
177         if(distanceToPreset <= SETPOINT_RADIUS) digitalWrite(4, 1);
178         else digitalWrite(4, 0);
179 
180         // Стабилизатор курса
181         double set_roll = constrain(1500 +
182                 HEADING_P*(courseToPreset - heading), 1350, 1650);
```

```
183
184         // Расчет управления на стабилизатор крена
185         double Koeff = 2.0;                 // коэффициент перевода градусов в ШИМ
186         pwmAil = constrain(1500 +
187                 Koeff*(map(set_roll, 1000, 2000, -20, 20)), 1350, 1650);
188
189         // Выдача ШИМ
190         AIL.writeMicroseconds(pwmAil);
191         /*****************************************************************************/
192         /**                              Конец разработки                           **/
193         /*****************************************************************************/
194
195         if (digitalRead(MODE_PIN) != mode)
196         {
197                 mode = digitalRead(MODE_PIN);
198                 Serial.print("mode = " + String(mode) + "\n");
199         }
200
201         // сохранение на SD карту
202         if (millis() - TLC_MS_SD > T_MS_SD)
203         {
204                 TLC_MS_SD = millis();
205                 sdWrite();
206         }
207 }
208
209 //============================================================================
210 // Функция настройки MT3333
211 void setupMt3333()
212 {
213   //--------------------------------------------------------------------------
214   // открытие последовательного порта
215   Serial1.begin(GPS_BAUD_RATE); while (!Serial1) {
216     ;
217   }
218
219 #ifdef GNSS_ENABLED
220   //--------------------------------------------------------------------------
221   // рассчет начального положения
222
223   // 1. ожидание спутников
224   Serial2.println("Searching sattelites");
225   unsigned long int gnssTimer = millis();
226   while ((nSats < N_SATS_MIN) && (millis() - gnssTimer < 60000))
227   {
228     delay(500);
229     while (Serial1.available() > 0)
230     {
231       if (gps.encode(Serial1.read()))
232       {
233         nSats = gps.satellites.value();
234         Serial2.print("nSats = ");
235         Serial2.println(nSats);
236       }
237     }
238   }
239
240   // 2. расчет среднего начального положения
241   int i = 0;
242   while (i < N_GNSS_0)
```

```
243   {
244     while (Serial1.available())
245     {
246       if (gps.encode(Serial1.read()))
247       {
248         initialLatitude += gps.location.lat();
249         initialLongitude += gps.location.lng();
250         Serial2.print(gps.location.lat()); Serial2.print("\t");
251         Serial2.print(gps.location.lng()); Serial2.print("\n");
252         i++;
253       }
254     }
255   }
256   initialLatitude /= float(N_GNSS_0);
257   initialLongitude /= float(N_GNSS_0);
258 
259   Serial2.print("Number of satellites found: ");
260   Serial2.print(nSats); Serial2.print("\n");
261   Serial2.print("Initial Latitude:  ");
262   Serial2.print(initialLatitude, 8); Serial2.print("\n");
263   Serial2.print("Initial Longitude: ");
264   Serial2.print(initialLongitude, 8); Serial2.print("\n");
265 #endif //GNSS_ENABLED
266 }
267 
268 //============================================================================
269 // Функция чтения MT3333
270 void readMt3333()
271 {
272   //----------------------------------------------------------------------
273   // Получение данных с ГНСС приемника
274   while (Serial1.available() > 0)
275   {
276     if (gps.encode(Serial1.read()))
277     {
278       // мгновенные замеры
279       hours = gps.time.hour();
280       minutes = gps.time.minute();
281       seconds = gps.time.second();
282       nSats = gps.satellites.value();
283       hdop = gps.hdop.value();
284       pfi = atoi(TGPSC_positionFixIndicator.value());
285 
286       // фильтруемые замеры
287       // 1. Заполнение массивов
288       shiftArray(latitudes, N_GNSS, gps.location.lat());
289       shiftArray(longitudes, N_GNSS, gps.location.lng());
290       shiftArray(altitudesMsl, N_GNSS, gps.altitude.meters() +
291                  atof(TGPSC_geoidSeparation.value()));
292       shiftArray(headings, N_GNSS,
293                  atof(TGPSC_heading_deg.value()) * DEG2RAD); //для mt3333
294       shiftArray(horizontalVelocities, N_GNSS,
295                  atof(TGPSC_horizontalVelocity_kh.value()) * KH2MS);
296 
297       //Serial2.println(gps.course.deg());
298 
299       // 2. фильтрация скользящим средним
300       latitude = mean(latitudes, N_GNSS);
301       longitude = mean(longitudes, N_GNSS);
302       altitudeMsl = mean(altitudesMsl, N_GNSS);
```

```
303         heading = mean(headings, N_GNSS);
304         horizontalVelocity = mean(horizontalVelocities, N_GNSS);
305 
306         // рассчитываеме параметры
307         velocityNorth = horizontalVelocity * cos(heading);
308         velocityEast = horizontalVelocity * sin(heading);
309       }
310     }
311   }
312 
313   //============================================================================
314   // функция записи результатов измерений на SD карту
315   void sdWrite()
316   {
317     //----------------------------------------------------------------------------
318     // формирование строки сообщения
319 
320     sdLogData = String(millis()) + "\t" +
321                 String(latitude, 10) + "\t" +
322                 String(longitude, 10) + "\t" +
323                 String(heading * RAD2DEG) + "\t" +
324                 String(mode) + "\t" +
325                 String(pwmAil) + "\n";
326 
327     //----------------------------------------------------------------------------
328     // запись сообщения
329     if (sdLog) sdLog.println(sdLogData);
330 
331     //----------------------------------------------------------------------------
332     // автосохранение данных
333     if (millis() - TLC_MS_AUTOSAVE > T_MS_AUTOSAVE)
334     {
335       sdLog.close();
336       SD.begin(SD_CHIP_SELECT_PIN);
337       sdLog = SD.open("log.txt", FILE_WRITE);
338       TLC_MS_AUTOSAVE = millis();
339     }
340   }
```

Задача 5.1.6. (7 баллов)

Написать программу для выделения контура и подсчета однотипных объектов (домов) на изображении. Объекты, находящиеся в кадре частично (не полностью), не должны определяться. Решением задачи является вывод изображения на экран с правильно определенными контурами объектов и отображение их количества.

Решение

Исходными данными являются:

1. Фотоизображения местности

Рис. 3 Исходное фотоизображение местности

2. Микроконтроллер Raspberry Pi Model 3B+
3. Установленное программное обеспечение, библиотеки и модули ОС Raspbian, Python 3, OpenCV, NumPY, Math.

Алгоритм выполнения задания:

1. Применить к изображению фильтры, убрать шумы. Параметры фильтрации подобрать самостоятельно.
2. Выделить все возможные контуры.
3. Выбрать из контуров те, которые удовлетворяют требованию задачи. Критерии отбора подобрать самостоятельно.
4. Отфильтровать выбранные контуры, убрать повторяющие

Код программы

```
import sys
import cv2
import numpy as np
if __name__ == '__main__':
    def nothing(*arg):
        pass
cv2.namedWindow( "result" )
cv2.namedWindow( "settings" )
img = cv2.imread("roof.png")
cv2.createTrackbar('h1', 'settings', 0, 255, nothing)
cv2.createTrackbar('s1', 'settings', 0, 255, nothing)
cv2.createTrackbar('v1', 'settings', 0, 255, nothing)
cv2.createTrackbar('h2', 'settings', 255, 255, nothing)
cv2.createTrackbar('s2', 'settings', 255, 255, nothing)
cv2.createTrackbar('v2', 'settings', 255, 255, nothing)
crange = [0,0,0, 0,0,0]
while True:
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h1 = cv2.getTrackbarPos('h1', 'settings')
    s1 = cv2.getTrackbarPos('s1', 'settings')
    v1 = cv2.getTrackbarPos('v1', 'settings')
    h2 = cv2.getTrackbarPos('h2', 'settings')
    s2 = cv2.getTrackbarPos('s2', 'settings')
    v2 = cv2.getTrackbarPos('v2', 'settings')
    h_min = np.array((h1, s1, v1), np.uint8)
```

```
    h_max = np.array((h2, s2, v2), np.uint8)
    thresh = cv2.inRange(hsv, h_min, h_max)
    cv2.imshow('result', thresh)
     ch = cv2.waitKey(5)
    if ch == 27:
        break
print(type(img))
cap.release()
cv2.destroyAllWindows()
```

Итоговый результат

```
import cv2
import numpy as np
if __name__ == '__main__':
   def callback(*arg):
       pass
cv2.namedWindow( "result" )
cv2.namedWindow( "result1" )
img = cv2.imread("roof.png")
#red
#2green
#3circule
hsv_min1 = np.array((0, 62, 170), np.uint8)
hsv_max1 = np.array((255, 124, 239), np.uint8)
hsv1 = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
thresh = cv2.inRange(hsv1, hsv_min1, hsv_max1)
st1 = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3), (1,1))
#st2 = cv2.getStructuringElement(cv2.MORPH_RECT, (11,11), (5,5))
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, st1)
#thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, st2)
contour1, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
#cv2.drawContours(img, contour1, -1, (0, 255, 0), 2)
contour=[]
if contour1:
    for cnt in contour1:
        moment = cv2.moments(cnt, 1)
        dm01 = moment['m01']
        dm10 = moment['m10']
        darea = moment['m00']
        if darea >= 900:
            x = int(dm10/darea)
            y = int(dm01/darea)
            contour.append(cnt)
            # cv2.circle(img, (x, y), 5, (0, 0, 255), 2)
            # cv2.putText(img, 'x: '+str(x)+'  '+'y: '+str(y), (x+10, y+10),
            # cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
cv2.drawContours(img, contour, -1, (0, 255, 0), 2)
print(len(contour))
while True:
    cv2.imshow('result', img)
    cv2.imshow('result1', thresh)
    ch = cv2.waitKey(5)
    if ch == 27:
        break
    cv2.destroyAllWindows()
```