```
230             }
231         }
232 
233         private void run() throws IOException {
234             in = new FastScanner(System.in);
235             out = new PrintWriter(System.out);
236 
237             init();
238             fillTime();
239             solve();
240 
241             out.flush();
242             out.close();
243         }
244 
245         public static void main(String[] args) throws IOException {
246             new Main().run();
247         }
248     }
```

§4 Заключительный этап: командная часть

Постановка задачи

Осуществить транспортную перевозку “груза” по суше, морю и воздуху, ориентируясь на показания датчиков, сигналы объектов инфраструктуры транспортной системы и данные внешней навигации. Команда получает оценку за совокупность результативных решений и может выбирать любое их сочетание и порядок реализации. Для каждой среды участникам предоставляется комплект конструктивных решений и компонентов (базовых наборов) с помощью и в рамках которых необходимо вести работу.

Базовые наборы для конструирования:

- Набор для конструирования беспилотного автомобиля
- Набор для конструирования беспилотного корабля
- Набор для конструирования квадрокоптера

Инструментарий

- Ноутбук, предоставляемый организаторами один на команду. Можно использовать свои ноутбуки в неограниченных количествах
- Паяльная станция – 1 на команду
- Элементы питания
- Набор ручного инструмента (пинцет, кусачки, плоскогубцы, макетный нож, набор отверток)
- Комплект расходных материалов (флюс, припой, изолента, пенопласт)

Программное обеспечение

- Среда программирования Arduino IDE
- Браузер
- Acrobat Reader (или аналог)
- QGroundControl
- PuTTy
- Pycharm community edition
- Python 2.7

Все дополнительные информационные материалы, схемы для сборки и тестовые библиотеки, предлагаемые участникам, доступны по ссылке: https://goo.gl/LLpraQ.

Элементы инфраструктуры:

- “Светофор” - устройство, сигнализирующее о возможности движения (имеет два управляющих состояния - “стоп”, “движение”).
- “Маяк” - устройство, сигнализирующее о пути следования корабля (имеет два управляющих состояния - “1”, “2”).
- “Порт погрузки” - зона полигона, из которой стартует транспортное средство, после того как на него загружен “груз” и дан разрешающий сигнал “светофора” или “маяка”.
- “Порт разгрузки” - зона полигона, оборудованная системой стыковки, в которой должно финишировать транспортное средство для разгрузки.
- “Разгрузочный кран” - механизм на границе двух сред, который перемещает “груз” с одного транспортного средства на другое (позиция погрузки и разгрузки точно зафиксированы).
- “Груз” - шайба, размером D80xH20, с намагниченными поверхностями.
- “Звездное небо” - система графических изображений над полигоном, которая позволяет получать точные координаты транспортного средства- X, Y, Θ - *угол (направление),* при помощи камеры, установленной на нем.

Подведение итогов

Итоговый результат определяется суммированием лучших попыток в каждом из трех треков общей задачи. В каждом треке можно заработать 50 баллов. Команда может заработать 10 дополнительных баллов в рамках трека “Автонет” после получения 90 баллов в рамках любых других задач. Также команда может заработать 50 баллов в рамках решения общих задач полигона АТС. Максимальное количество баллов - 210. Количество зачетных попыток ограничено (каждый день - не более пяти по каждому подтреку, но при использовании каждой последующей попытки одного и того же задания, результат последующей попытки уменьшается на 0.5 балла).

Задача	Максимальный балл
“Маринет”	50
“Автонет”	50
“Аэронет”	50
АТС	50
Доп. Баллы “Автонет”	10
Итого	210

Важно! Каждая команда должна в конце каждого рабочего дня сдать все последние версии файлов с программным кодом в формате:

Файл должен называться по названию подтрека и иметь в названии номер версии, например "Marinet_v1.ino". Кроме того, в блоке комментариев в заголовке программы необходимо указать название команды, ФИО участника/участников, версию (номер попытки) и название команды.

Тренировки

Подход к стенду свободный, если нет других желающих. Иначе каждой команде даётся 5 минут на тестирование.

Маринет

Задача
В треке по беспилотным плавательным устройствам вам нужно будет собрать и запрограммировать корабль с дифференциальным приводом (двумя отдельно управляемыми гребными колесами), который должен преодолеть заданный водный маршрут, используя доступные инструменты вместе или на выбор (данные УЗ-датчиков, навигационные данные, лазерный дальномер) и доставить груз в порт разгрузки таким образом, чтобы он попал точно в заданный сектор работы разгрузочного крана.

Плавательное средство начинает свою работу получив сигнал от “маяка” (ИК светофора). Сигнал бывает 2 типов и он определяет путь движения транспортного средства.

Требования к механике:

“Корабль” должен держаться на воде без перекосов более 5 мм. Лопасти собранного корабля (со всей нагрузкой) не должны цепляться за дно бассейна. При стыковке (магнитной) с портом груз возможно снять погрузочным краном. Конструкция должна удобно собираться и разбираться.

Базовый набор:

- Комплект для сборки «корабля» с системой крепления приводов, сенсоров, груза, системы питания, система стыковки к порту, системой навигации.
- Управляющая плата Arduino
- Моторы - приводы гребных колес
- Raspberry pi с камерой для навигации
- Комплект электронных компонентов
- Комплект крепежных элементов

Образец для сборки корабля предоставляется организаторами.

Схема полигона

Полигон представляет собой бассейн размером 1.5м на 2.7м, наполненный водой и установленными блоками.

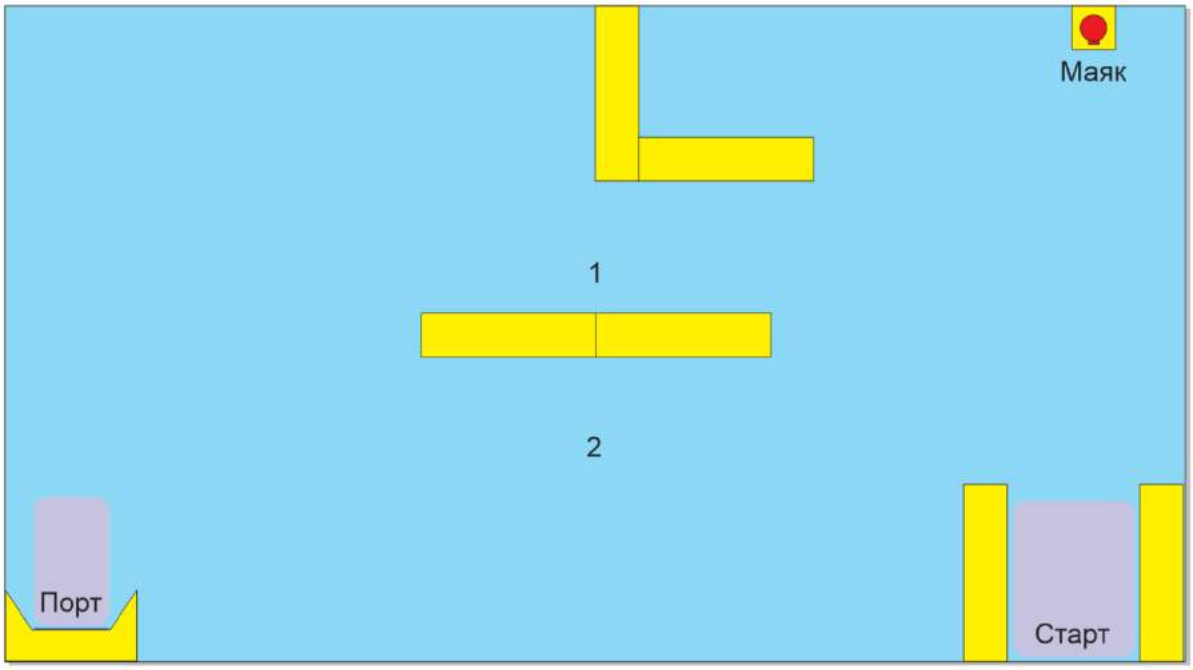


Фото полигона

После получения сигнала светофора корабль начинает выполнение задания - движение к порту. Левый сигнал - путь через ворота номер два. Правый сигнал - путь через ворота номер один.

Баллы

	Задача	Баллы
1	Сборка устройства (*задание принимается до окончания второго соревновательного дня)* Электроника Пройти отладочное тестирование основных узлов: 1. Демонстрация работы лопастей Лопасти вращаются вперед, назад и в разные стороны (1 балл). 2. Демонстрация работы ИК приемника При включении левого сигнала светофора на корабле загорается левый сигнальный светодиод. При включении правого сигнала светофора на корабле загорается правый сигнальный светодиод (1 балл). 3. Демонстрация работы контроля напряжения На экран компьютера выводится напряжение питания системы (1 балл). 4. Демонстрация работы навигационной системы На экран компьютера выводится результат работы тестовой программы - координаты и угол (1 балл). Механика 1. Собрать корабль, держащийся на воде без перекосов более 5 мм, не цепляющий лопастями за дно бассейна по всей площади, несущий на себе всю электронику, наличие оборудования для крепления к порту (1 балл). 2. При стыковке (магнитной) в порт груз возможно снять краном (1 балл).	0-6
2	Выход из зоны старта (*задание принимается до окончания второго соревновательного дня)*	3

	После срабатывания светофора, корабль должен начать движение. Задание считается выполненным, если корабль пересекает линию старта (по границе блока, ближнего к светофору) Штрафы: • За касание дна лопастями - 1 балл Дополнительно: Предварительно требуется выполнить задание 1 “Собрать корабль”.	
3	Необходимо встать на линию маяка, получить его сигнал от левого излучателя и начать движение. Необходимо доплыть до маяка, то есть, коснуться его основания.	5
4	Доплыть до порта без препятствий и выбора пути и остановиться. Пересечение центральной линии (5 баллов). Швартовка в порт кормовой частью корабля (5 баллов). Остановка двигателей (2 балла).	12
5	Выполнить задание Получить сигнал от маяка, доплыть до порта по заданному пути и остановиться. При получении сигнала от левого излучателя плыть надо по пути 1. При получении сигнала от правого излучателя, нужно плыть по пути №2. Запарковаться в порту ровно под стрелой крана с возможностью разгрузки груза, остановить двигатели, зажечь оба сигнальных светодиода. Путь определяется жюри случайным образом перед стартом, непосредственно после установки корабля на полигоне. Штрафы: • Груз не находится под стрелкой крана - 5 баллов • Движение не в тот тоннель - 5 баллов • Не выключил двигатели в порту - 3 балла • Касание борта или блока - 1 балл • Столкновение с маяком - 1 балл	19

6	Сразу же после выполнения задания 4 продемонстрировать движение по пути, не выпавшему в 4 задании (изменять конструкцию устройства и прошивку не разрешается)	5
	Всего баллов:	50

Штрафные баллы:

- Касание борта, волнореза, буйка - штраф 1 балл
- Столкновение с маяком - штраф 1 балл
- За использование второй и последующих попыток – $Ш = К - 0{,}5 \times (N - 1)$, где Ш – штрафные баллы, К –количество баллов за данное задание, N – номер попытки

Вариант сборки устройства:

Полное решение:

```
#define __AVR_ATmega168__ 1
#include <ros.h>
#include <geometry_msgs/Pose2D.h>
#include <IRLibDecodeBase.h>
#include <IRLib_P02_Sony.h>
#include <IRLibCombo.h>
#include <Wire.h>
#include <VL53L0X.h>
```

```
#include <IRLibRecv.h>
IRrecv myReceiver(A3);  //Пин куда подключен сигнал приемника
IRdecode myDecoder;
int max_speed = 100;
#define IN1 10
#define IN2 11
#define IN3 5
#define IN4 9
VL53L0X lidar;
int lidarMax = 800;
int lidarTemp;
int lidarDist = lidarMax;
String str;
#define MAX_DISTANCE 200
#define FRsonar 8  // Передний правый
#define RRsonar 12 // Задний правый
#define FLsonar A4 // Передний левый
#define RLsonar A5 // Задний левый
#define Vin A0 // Контроль напряжения
#define Mins 50 // Минимальное расстояние до борта
double distFR;
double distRR;
double distFL;
double distRL;
float cord[3];
float Voltage;
char IK = 'N';
void setup() {
  delay(3000);
  Serial.begin(57600);
  Wire.begin();
  Serial1.begin(57600);
  pinMode(13, OUTPUT);
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  myReceiver.enableIRIn(); // Запуск ресивера
  //DC motor pins
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  lidar.init();
  lidar.setTimeout(200);
  lidar.startContinuous();
  IK = Receiver();
  while(IK == 'N') IK = Receiver();
  Cord();
  vertushka(-90.0);
  Cord();
  if (IK == 'L'){
    goHome(0.85, cord[1], -90.0, false);
    vertushka(-35.0);
    delay(2000);
    goHome(cord[0], 2.44, -3.0, true);
    Cord();
    vertushka(55.0);
    delay(2000);
    Cord();
    goHome(0.22, cord[1], 88.0, false);
  }
  else{
    goHome(0.9, cord[1], -85.0, false);
    vertushka(10.0);
    delay(200);
```

```
    goHome(cord[0], 0.9, 45, true);
    vertushka(0.0);
    delay(200);
    goHome(cord[0], 2.44, -2.0, true);
    Cord();
    vertushka(65.0);
    delay(200);
    Cord();
    goHome(0.22, cord[1], 80.0, false);
  }
  goMotor(0,0);
  digitalWrite(A1, HIGH);
  digitalWrite(A2, HIGH);
  goMotor(0,0);
  ////////////////////////////////
}
void loop() {
  //goMotor(117, 120);
}
void vertushka(double theta) {
  int dt = 2;
  while (abs(theta - cord[2]) >= dt)
  {
    Cord();
    if (cord[2] < theta)
      goMotor(-100, 80);
    else
      goMotor(80, -100);
      dt = 20;
      delay(10);
  }
}
void goHome(double X, double Y, double angle, bool y){
  double a[2] = {X,Y};
  Cord();
  while (abs(cord[y] - a[y]) > 0.04){
    if (cord[2] - angle >= 0.5){
      goMotor(100 + int(ceil(abs((cord[2] - angle))) * 3.3), 100);
    }
    else
    if (cord[2] - angle <= -0.5){
      goMotor(100, 100 + int(ceil(abs((cord[2] - angle))) * 4.0));
    }
    else
    goMotor(96, 100);
    Cord();
  }
  goMotor(-150, -150);
  delay(1450);
  goMotor(0,0);
  Cord();
}
void goMotor(int lft, int rgt ) {
  if (lft > 0) {
    analogWrite(IN2, lft);
    digitalWrite(IN1, 0);
  } else {
    analogWrite(IN2, 255 - abs(lft));
    digitalWrite(IN1, 1);
  }
  if (rgt < 0) {
    analogWrite(IN3, 255 - abs(rgt));
    digitalWrite(IN4, 1);
  } else {
```

```
    analogWrite(IN3, rgt);
    digitalWrite(IN4, 0);
  }
}
void SonarScan ()
{
  distFL = getSonar(FLsonar);
  Serial.print("FL:");
  Serial.println(distFL);
  distRL = getSonar(RLsonar);
  Serial.print("RL:");
  Serial.println(distRL);
  distFR = getSonar(FRsonar);
  Serial.print("FR:");
  Serial.println(distFR);
  distRR = getSonar(RRsonar);
  Serial.print("RR:");
  Serial.println(distRR);
}
double getSonar(int SonarPin) {
  pinMode(SonarPin, OUTPUT);
  digitalWrite(SonarPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(SonarPin, LOW);
  pinMode(SonarPin, INPUT);
  return pulseIn(SonarPin, HIGH, 15000.0) / 58.0;
}
char Receiver() {
  if (myReceiver.getResults())
  {
    myDecoder.decode(); // Декодируем...
    if (myDecoder.protocolNum == SONY)  {
      Serial.println(myDecoder.value, HEX);
    }
    myReceiver.enableIRIn();   // Перезапускаем приемник
    if (myDecoder.value == 0x60) {
      digitalWrite(A2, HIGH);  // Правый сигнальный светодиод
      digitalWrite(A1, LOW);
      return 'R';
    }
    if (myDecoder.value == 0x50) {
      digitalWrite(A1, HIGH);   // Левый сигнальный светодиод
      digitalWrite(A2, LOW);
      return 'L';
    }
    return 'N';
  }
  return 'N';
}
void GetV() {
  double val = analogRead(Vin);
  val = ((val * 4.9) / 1024.0) * 2.0;
  Serial.print("volts: "); Serial.println(val);
}
void Cord() {
  while (Serial1.available() > 0) {
    char recieved = Serial1.read();
    str += recieved;
    if (recieved == '\n') {
      //Serial.print(str);
      int index = str.indexOf(';');
      int secondIndex = str.indexOf(';', index + 1);
      // put your main code here, to run repeatedly:
      cord[0] = str.substring(0, index).toFloat();
```

```
      cord[1] = str.substring(index + 1, secondIndex).toFloat();
      cord[2] = str.substring(secondIndex + 1).toFloat();
      Serial.println(cord[0]);
      Serial.println(cord[1]);
      Serial.println(cord[2]);
      Serial.println("\n");
      str = "";
    }
  }
}
```

Автонет

Задача

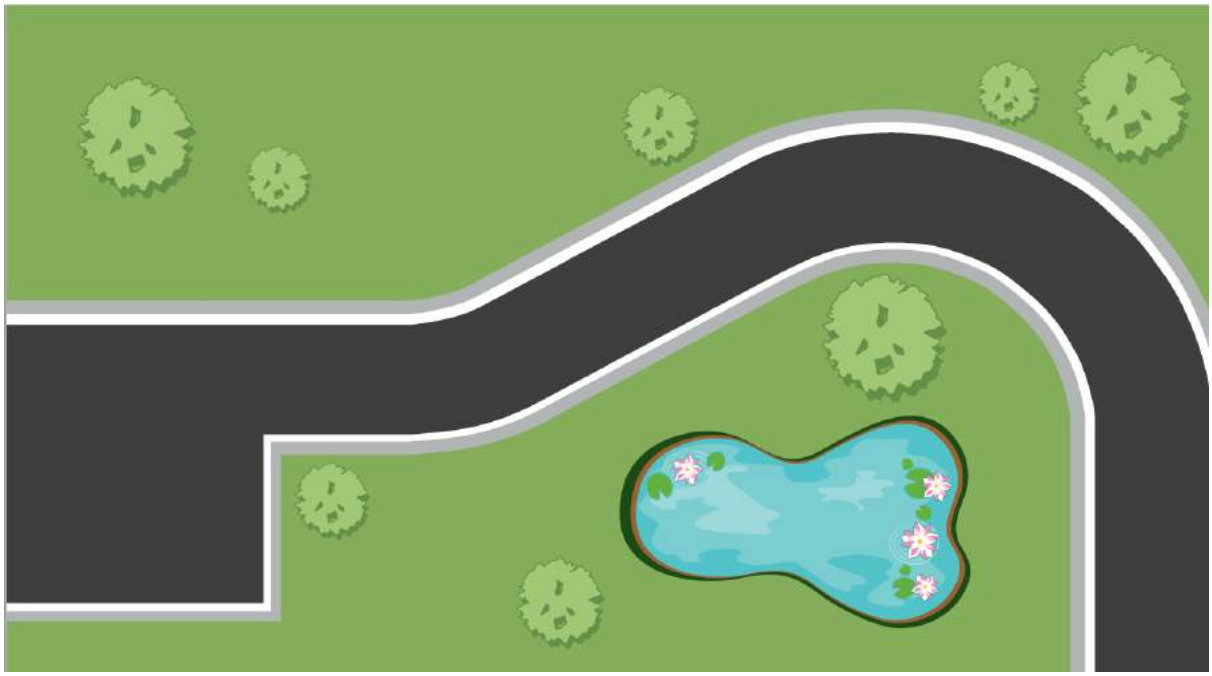
В задаче по беспилотным автомобилям необходимо разместить электронику и сенсоры на четырехколесном шасси (с автомобильной кинематикой) для решения задачи автономного доставка груза из порта отгрузки в аэропорт, ориентируясь по координатам системы навигации, элементам инфраструктуры и сигналам ИК-светофора.

Базовый набор:

- Шасси с мотором, регулятором хода и рулевым сервоприводом
- Зарядное устройство
- Управляющая плата Arduino
- Raspberry pi
- Набор датчиков
- Комплект крепежных элементов
- Навигационная карта
- Набор для сборки ИК приемника

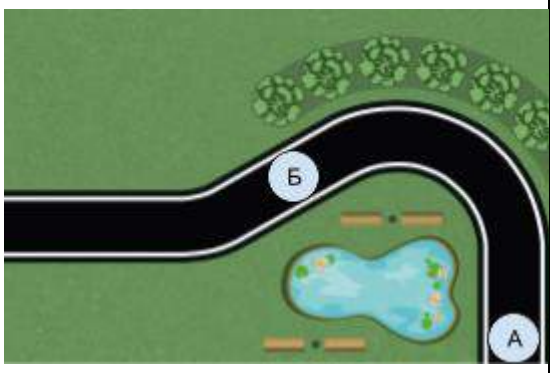
Схема полигона

Баннерное покрытие с нанесенной разметкой (размер: 1.5м х 2,7м)



Баллы

		Задача	Баллы
1. Первый раздел (1-12)			
1	Проехать прямо в течении 2 секунд, поехать назад в течении 2 сек Выход из зоны старта (*задание принимается до окончания второго соревновательного дня)*	Машина полностью собрана: все датчики электрически подключены правильно. Спаять кабель питания для для Raspberry. Собрать ИК приемник для работы со светофором. Автомобиль может ехать вперед 2 секунды и назад 2 секунды.	*10*
2	*Ехать по кругу по часовой стрелке 2 секунды, против часовой 2 секунды Выход из зоны старта (*задание принимается до окончания второго соревновательного дня)*	Машинка должна непрерывно ехать по часовой стрелке в течении 2 секунд и затем против часовой стрелки в течении 2 секунд.	*2*
3	Вывести в ком порт данные с УЗ дальномера	Вывести на экран получаемую с помощью УЗ датчика информацию о расстоянии до препятствия в см	*1*
4	Ехать прямо, пока не появится препятствие. Остановиться перед препятствием. Остановиться навсегда.	Выполнить остановку машины перед препятствием	*1*
5	*“Прилипала”. Двигаться вперед или назад, всегда находясь на расстоянии 30 см. от препятствия.	Следовать за предполагаемым транспортным средством, повторяя его движения по прямой и в обратном направлении	*3*
6	Выводить в ком порт сигнал светофора	Получить данные от ИК приемника и вывести данные в ком-порт	*1*

7	*Останавливаться и ехать вперед по сигналу светофора.	Получить сигнал от ИК приемника и выполнить действие (не двигаться по сигналу “левый”/движение по сигналу “правый”)	*2*
8	Выводить в com порт текущие координаты машинки (X,Y, угол).	Выводить в com порт текущие координаты машинки (и направление).	*1*
9	*Проехать по прямой по меткам	Использовать навигационную карту для движения по прямой. Машина ставится под произвольным углом по отношению к линии движения на старте.	*3*
10	Проехать из пункта А (порт 1) в пункт Б (конец поворота)	Использовать навигационную карту для проезда поворота, не выезжая за разметку (белые линии не пересечены корпусом автомобиля). Машина ставится под произвольным углом по отношению к линии движения на старте. 	*5*
11	*Проехать из пункта А (порт 1) в пункт Б (место разгрузки)	Использовать навигационную карту для проезда трассы, не выезжая за разметку. Машина ставится под произвольным углом по отношению к линии движения на старте. Старт осуществляется по сигналу ИК светофора. Штраф за нарушение сигнала светофора - 1 балл	*15*

		Штраф за нарушение разметки (более 2 колес) - 1 балл Штраф за невозможность взять груз краном - 5 баллов  	
12	*Проехать из пункта А в место разгрузки, реагируя на внешние условия различных типов.	Остановка перед препятствием	*6*
		Всего баллов:	*50*
Второй раздел. Дополнительные задания. Компьютерное зрение (Дополнительные 1-3) *Приступать к выполнению дополнительных заданий можно набрав не менее 90 баллов по остальным заданиям.*			
Дополнительное 1	Детектирование дорожного знака на изображении.	Определить положение дорожного знака на изображении, получаемом из камеры, выделив знак в рамку.	*1*
Дополнительное 2	*Отличить один знак от другого. Дополнить алгоритм распознаванием знаков.	Из предложенного набора знаков выбрать тот, который наиболее соответствует знаку, распознанному на изображении, получаемом из камеры (2 балла), дополнить алгоритм проезда трассы функцией распознавания знаков (2 балла).	*4*

Дополнительное 3	*Определить наличие пешехода на изображении.	Определить, есть ли пешеход на изображении и сообщить о его наличии произвольным сигналом	*5*
		Всего баллов:	*10*

Максимальное количество баллов за все задания: 60

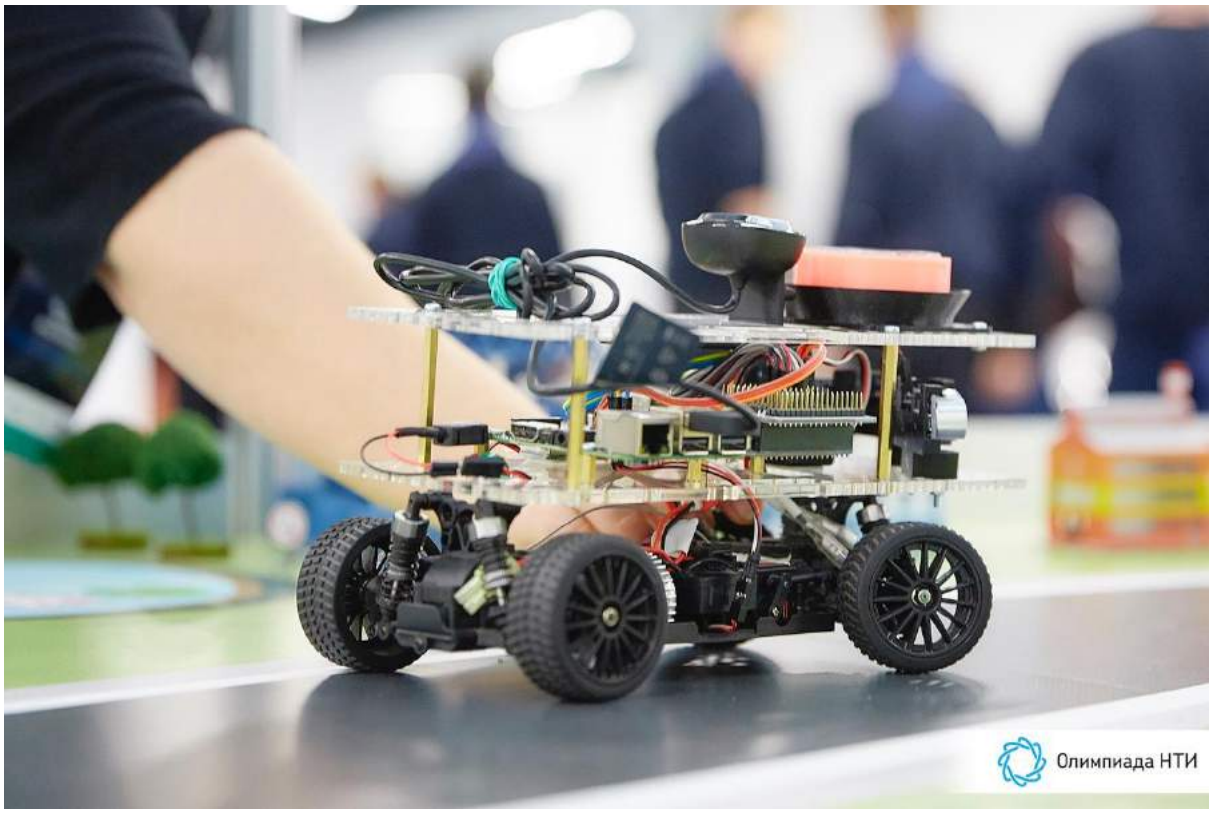
Задание состоит из двух разделов. Первый раздел включает в себя простые задания для подготовки к проезду полигона. Второй раздел включает дополнительные задания, основанные на методах компьютерного зрения, используемых в современных беспилотных автомобилях. Разделы заданий могут быть выполнены частично или полностью. Возможно выполнить задания второго, дополнительного, раздела только после того, как команда заработает 90 баллов в рамках любых подтреков. За выполнение всех заданий дополнительного раздела можно в сумме получить не более 10 баллов. При выполнении задания со звездочкой, выделенного жирным шрифтом, также засчитываются баллы предыдущих заданий этой же категории (задания в списке от предыдущего задания со звездочкой до выполненного).

Штрафные баллы:

- За разрушение дома и других элементов полигона (сдвинут с места) - 2 балла
- За нарушение разметки (более 2 колес) - 1 балл
- За нарушение сигнала светофора - 1 балл
- За использование второй и последующих попыток – Ш=К-0,5×(N-1), где Ш – штрафные баллы, К –количество баллов за данное задание, N – номер попытки

В любых проездах разрешается выезжать одновременно максимум двумя колесом за полосу.

Вариант сборки устройства:



Полное решение:

```
#include <NewPing.h>
#include <Servo.h>
int Mid = 90;
int Left = 110;
int Right = 70;
//1320
int Forv = 1300; //1300 //Вообще говоря это назад. Именно, что бы поехать вперёд нужно сначала дать стоп.
#define Stop 1500
#define Back 1680
//NewPing LeftSon(26, 28, 70);
NewPing MidSon(22, 24, 70);
//NewPing RightSon(30, 32, 70);
Servo Motor;
Servo Serv;
//////Глобальные переменные для приёма сигнала с распбери
int x = 5;
int y = 5;
int angle = -90; //Координаты в см и угол в градусах
int znak = 0;
int spd = 1500;   //
int Empty = 0;
////////Вывод текущих координат в сериал порт
void PrintCoordinate(void)
{ Serial.println();
  Serial.print("X=");
  Serial.print(x);
  Serial.print(" :: Y=");
  Serial.print(y);
  Serial.print(" :: Angle=");
  Serial.println(angle);
}
int State = 1;
```

```
//          1 Можно ехать
//          2 Нет сигнала от распберри
//          3 Не видно маркер
//          4 Сонар видит препятствие
//          5 Светофор запрещает ехать
// Читает String от Raspberry и вытаскивает из неё координаты и угол
// координаты и угол хранятся в глобальных переменных
// Если координаты прочитаны успешно, то возвращает true
// Если камера не видит маркер, функция возвращает false
bool GetCoordinate()
{ if (State == 1)
  { Motor.writeMicroseconds(Stop);
  }
  //delay(10);
  String str = Serial3.readStringUntil('\n');
  //  Serial.print("str : ");
  //  Serial.println(str);
  //str = "00/123/456/-179";
  if (str.substring(0, 2).toInt() == -5)
  { Serial.println("Marker not found");
    return false;
  }
  x = str.substring(3, 6).toInt();
  y = str.substring(7, 10).toInt();
  angle = str.substring(11, str.length()).toInt();
  PrintCoordinate();
  if (State == 1 )
  { Motor.writeMicroseconds(Forv);
  }
  return true;
}
void setup()
{
  Serial.begin(9600);
  Serial3.begin(9600);
  Motor.attach(17, 800, 2300);
  Serv.attach(16);
  spd = 1500;
  Motor.writeMicroseconds(Stop);
  Serv.write(Mid);
  delay(1500);
}
void Revers()
{
  Serv.write(Mid);
  Motor.writeMicroseconds(2000);
  delay(100);
  Motor.writeMicroseconds(Stop);
  delay(40);
  Motor.writeMicroseconds(Forv);
  delay(40);
  Motor.writeMicroseconds(Stop);
  delay(40);
  Motor.writeMicroseconds(Forv);
  delay(30);
  Motor.writeMicroseconds(Stop);
}
void ReversEnd()
{
  Serv.write(Mid);
  Motor.writeMicroseconds(2000);
  delay(25);
  Motor.writeMicroseconds(Stop);
  delay(40);
```

```
  Motor.writeMicroseconds(Forv);
  delay(40);
  Motor.writeMicroseconds(Stop);
  delay(40);
  Motor.writeMicroseconds(Forv);
  delay(30);
  Motor.writeMicroseconds(Stop);
}
void Line()
{ int Mid = 98;
  int Left = 105;
  int Right = 75;
  while (Serial3.available() == 0)
  {
    Serial.println("Endle");
  }
  while ( x < 40 && y < 50 ) //Первый прямой участок
  { if (Serial3.available())
    { Empty = 0;
      if (GetCoordinate())
      { State = 1;
        Serial.println("1");
        //if (IRRECEIVER)
        //        if ((MidSon.ping_cm() > 1) && (MidSon.ping_cm() < 15))
        //        { State = 4;
        //        }
      }
      else
      { State = 3;
      }
    }
    else  // Сериал порт пустой от Raspberry
    { Serial.println("Serial is empty");
      //State = 1;
      Empty++;
      if (Empty > 6) //Сериал порт пустой давно
      { Empty = 6;
        State = 2;
      }
    }
    // Если можно ехать - рулю. Если нельзя торможу.
    Serial.print("State : ");
    Serial.println(State);
    if (State == 1)
    { Motor.writeMicroseconds(Forv);
      //      delay(20);
      //      Motor.writeMicroseconds(Stop);
      if (y > 14)
      { Serv.write(Right);
        Serial.println("RIGHT");
      }
      else
      { if (y < 10 )
        { Serv.write(Left);
          Serial.println("LEFT");
        }
        else
        { Serv.write(Mid);
          Serial.println("MID");
        }
      }
      //      if (Angle>-94 && Angle <-86)
      //      {Serv.write(Mid);
      //          Serial.println("MID");}
```

```
    }
    else
    { Motor.writeMicroseconds(Stop);
    }
    Serial.println();
  }
  Revers();
  if (2400 > (x - 45) * (x - 45) + (y - 75) * (y - 75))
  {
    //      Serv.write(Right);
    //      Serial.print("RIGHT");
    Serv.write(Mid);
    Serial.print("MID");
  }
  else
    //59*59=3481  //58*58=3364
  { if (2600 < (x - 45) * (x - 50) + (y - 75) * (y - 75))
    { Serv.write(Left);
      Serial.print("LEFT");
    }
    else
    { Serv.write(Mid);
      Serial.print("MID");
    }
  }
  //Motor.writeMicroseconds(Stop);
  //delay(300);
//  while (1)
//  { Serial.println("EndLine");
//    Serv.write(Mid);
//    Motor.writeMicroseconds(Stop);
//  }
}
void Circle()
{
  Mid = 90;
  Left = 120;
  Right = 60;
  //Forv = 1250;
  while ( y < 75 ) //Окружность
  { Serial.println("Circle");
    if (Serial3.available())
    { Empty = 0;
      if (GetCoordinate())
      { State = 1;
        //if (IRRECEIVER)
        //        if ((MidSon.ping_cm() > 1) && (MidSon.ping_cm() < 15))
        //        { State = 4;
        //        }
      }
      else
      { State = 3;
      }
    }
    else  // Сериал порт пустой от Raspberry
    { //State = 1;
      Serial.println("Serial is empty");
      Empty++;
      if (Empty > 6) //Сериал порт пустой давно
      { Empty = 100;
        State = 2;
      }
    }
    // Если можно ехать - рулю. Если нельзя торможу.
```

```
    Serial.print("State : ");
    Serial.println(State);
    if (2400 > (x - 45) * (x - 45) + (y - 75) * (y - 75))
    {
      //      Serv.write(Right);
      //      Serial.print("RIGHT");
      Serv.write(Mid);
      Serial.print("MID");
    }
    else
      //59*59=3481  //58*58=3364
    { if (2600 < (x - 45) * (x - 50) + (y - 75) * (y - 75))
      { Serv.write(Left);
        Serial.print("LEFT");
      }
      else
      { Serv.write(Mid);
        Serial.print("MID");
      }
    }
    if (State == 1)
    { Motor.writeMicroseconds(Forv);
      //      delay(50);
      //      Motor.writeMicroseconds(Stop);
    }
    else
    { Motor.writeMicroseconds(Stop);
    }
    Serial.println();
    Serial.println();
  }
  Revers();
//  while (1)
//  { Serial.println("EndCircle");
//    Motor.writeMicroseconds(Stop);
//    Serv.write(Mid);
//  }
}
void Line2()
{ int Mid = 90;
  int Left = 120;
  int Right = 60;
  //1250;
  //Forv=1500;
  while ( y < 150 && x>55 ) //Второй прямой участок
  { Serial.println("Line2");
    if (Serial3.available())
    { Empty = 0;
      if (GetCoordinate())
      { State = 1;
        Serial.println("1");
        //if (IRRECEIVER)
        //        if ((MidSon.ping_cm() > 1) && (MidSon.ping_cm() < 15))
        //        { State = 4;
        //        }
      }
      else
      { State = 3;
      }
    }
    else  // Сериал порт пустой от Raspberry
    { Serial.println("Serial is empty");
      //State = 2;
      Empty++;
```

```
      if (Empty > 6) //Сериал порт пустой давно
      { Empty = 6;
        State = 2;
      }
    }
    // Если можно ехать - рулю. Если нельзя торможу.
    Serial.print("State : ");
    Serial.println(State);
    if (State == 1)
    { Motor.writeMicroseconds(Forv);
      //      delay(75);
      //      Motor.writeMicroseconds(Stop);
      if (y < 282 - 1.9 * x)
      { Serv.write(Right);
        Serial.println("RIGHT");
      }
      else       //285
      { if (y > 288 - 1.9 * x )
        { Serv.write(Left);
          Serial.println("LEFT");
        }
        else
        { Serv.write(Mid);
          Serial.println("MID");
        }
      }
      if (angle < 71 && angle > 60)
      { Serv.write(Mid);
        Serial.println("MID");
      }
    }
    else
    { Motor.writeMicroseconds(Stop);
    }
    Serial.println();
  }
  Revers();
  //delay(300);
    while (1)
    { Serial.println("EndLine2");
      Motor.writeMicroseconds(Stop);
    }
}
void End()
{ int Mid = 90;
  int Left = 120;
  int Right = 60;
  //int Forv = 1305; //1250;
  while ( y < 252 && x > 34 ) //Второй прямой участок
  { Serial.println("End");
    if (Serial3.available())
    { Empty = 0;
      if (GetCoordinate())
      { State = 1;
        Serial.println("1");
        //if (IRRECEIVER)
        //        if ((MidSon.ping_cm() > 1) && (MidSon.ping_cm() < 15))
        //        { State = 4;
        //        }
      }
      else
      { State = 3;
      }
    }
```

```
    else  // Сериал порт пустой от Raspberry
    { Serial.println("Serial is empty");
      //State = 2;
      Empty++;
      if (Empty > 6) //Сериал порт пустой давно
      { Empty = 6;
        State = 2;
      }
    }
    // Если можно ехать - рулю. Если нельзя торможу.
    Serial.print("State : ");
    Serial.println(State);
    if (State == 1)
    { Motor.writeMicroseconds(Forv);
      //delay(50);
      //Motor.writeMicroseconds(Stop);
      if (y < 372 - 3.31 * x)//(y < 313 - 1.5 * x)
      { Serv.write(Right);
        Serial.println("RIGHT");
      }
      else
      { if (y > 376 - 3.31 * x)//(y > 313 - 1.5 * x )
        { Serv.write(Left);
          Serial.println("LEFT");
        }
        else
        { Serv.write(Mid);
          Serial.println("MID");
        }
      }
    }
    else
    { Motor.writeMicroseconds(Stop);
    }
    Serial.println();
  }
  ReversEnd();
  while (1)
  { Serial.println("EndAll");
    Motor.writeMicroseconds(Stop);
  }
}
void loop()
{ Serial.println("Start");
  Line();
  Circle();
  Line2();
  //  Line3();
  End();
}
```

Аэронет

Задача

В задаче необходимо выполнить автономную доставку груза, управляя квадрокоптером автономно, при помощи Arduino и навигации по ArUco – маркерам (см. фото полигона).

Описание задачи: необходимо осуществить автономную доставку груза в рамках заданной траектории.

Траектория доставки:

- точки А - старт (прием груза). Беспилотный летательный аппарат (здесь и далее – БПЛА) устанавливается на точку А (приема груза) перед началом соревнований. Необходимо получить сигнал от ИК светофора о начале задания.
- точка Б - место сброса груза
- точка В - место посадки БПЛА

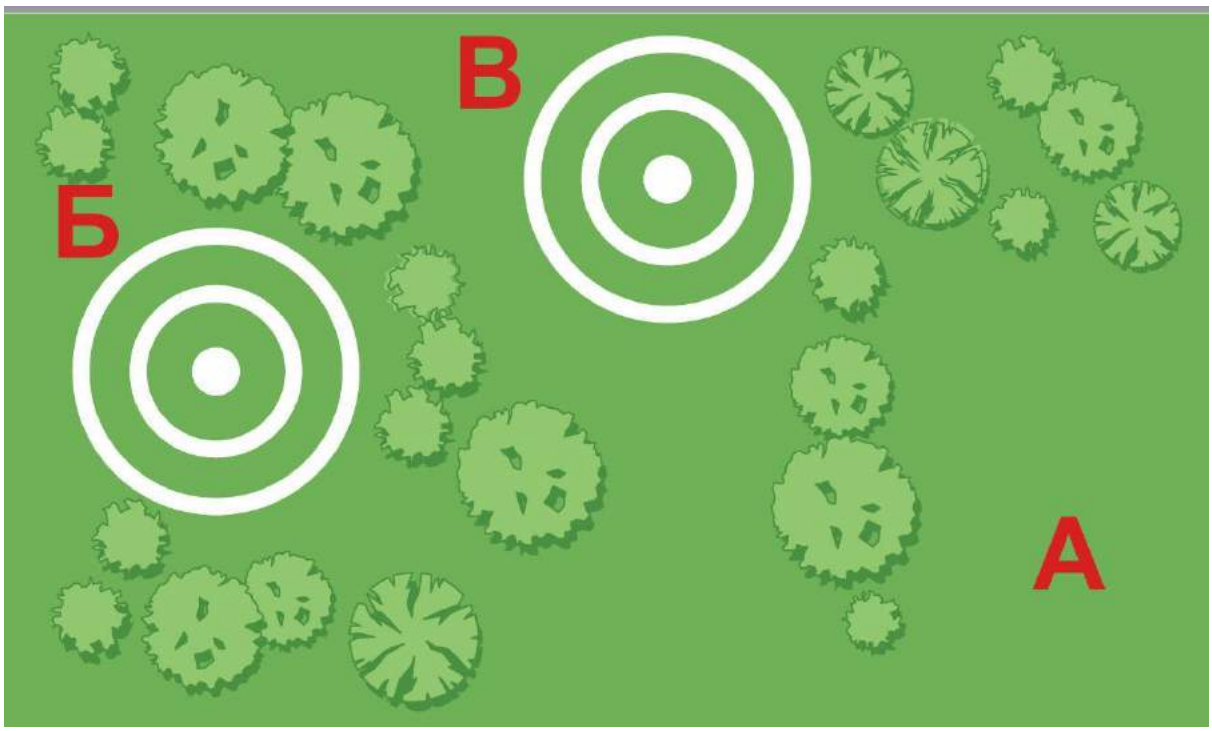
До проведения испытаний на полигоне Участникам необходимо:
1.1 Установить захватное устройство на БПЛА
1.2 Запрограммировать захватное устройство
1.3 Проверить работоспособность захватного устройства
1.4 Подключить светодиодную ленту
1.5 Индикация ленты демонстрирует состояние захвата
1.6 Индикация отображает полетное состояние БПЛА (взлет, зависание, перелет или посадку)
1.7 Провести предполетную проверку БПЛА (Приложение А)
1.8 Загрузить тестовый скетч по зависанию над меткой (Takeoff_example.ino)
На полигоне (выполняется экспертом):

2.1 Проверить летные характеристики БПЛА, используя визуальные характеристики
2.2 Проверить работоспособность kill switch (при переключении стика SwA на пульте в нижнее положение моторы останавливаются)
2.3 Проверить наличие полетных режимов
2.4 Проверить зависание над меткой
2.5. Проверить ИК датчик (прием сигнала от стартовой метки)
Рекомендации:
пп.1.7-1.8 желательно выполнить в первую очередь, чтобы сразу же перейти к п.2.1 - 2.5, параллельно можно выполнять п.1.1-1.6.

Базовый набор:

- Конструктор программируемого квадрокоптера Clever
- Управляющая плата Arduino
- Raspberry pi с камерой для навигации
- Адресная светодиодная лента
- Захватное устройство
- Комплект электронных компонентов
- Комплект крепежных элементов

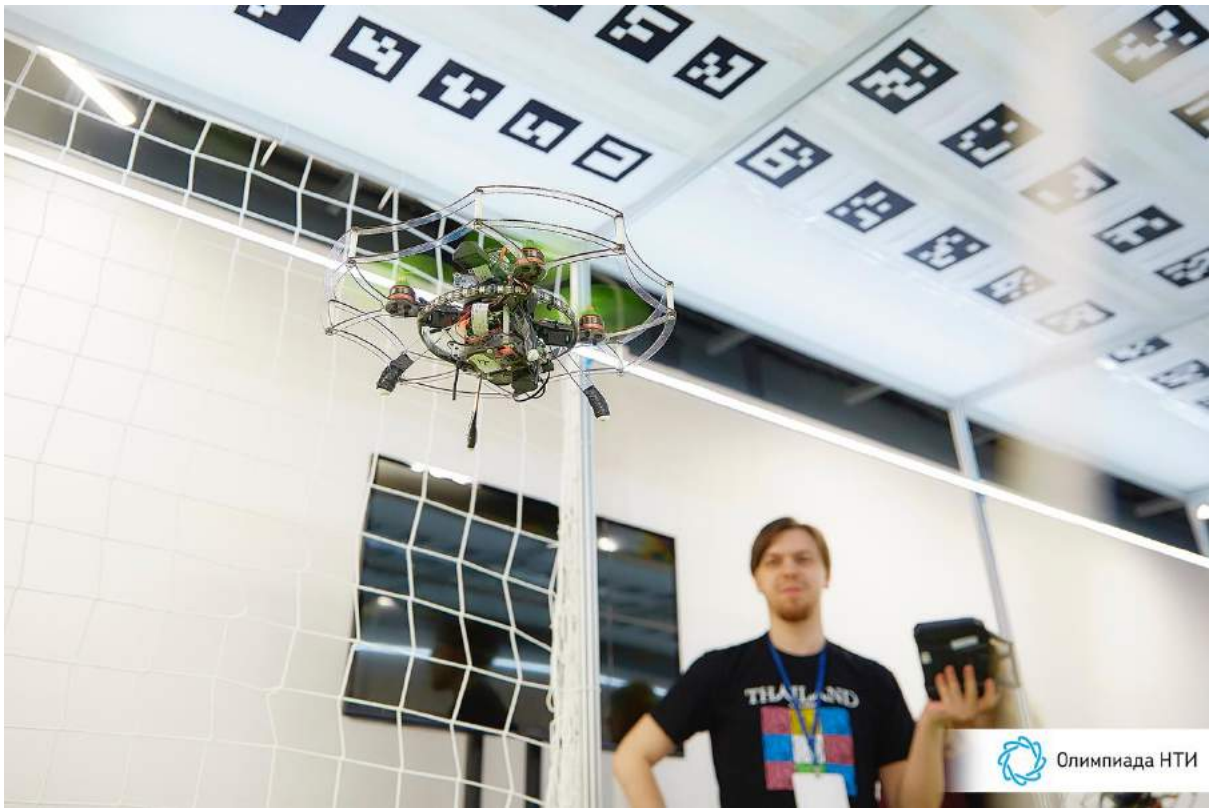
Схема полигона



Стенд представляет собой ограниченное пространство (2.74х1.52х1.4 м (ДхШхВ)).

Стены – сетка, сверху – система, на полу расположено препятствие, область сброса груза (Б) и область посадки (В) в виде кругов.

Фото полигона



Баллы

№	Задание	Баллы
1	Сборка захвата. Захват собран и установлен на коптере (2 балла), функционирует корректно (2 балла)	4
2	Индикация ленты демонстрирует состояние захвата.	3

3	Индикация отображает полетное состояние БПЛА (взлет, зависание, перелет или посадку), при этом отображаются два и более состояний (3 балла). Штрафы: • Отображается одно состояние – 2 балла	3
4	Автономный взлет. Оторваться от поверхности на 30 см.	2
5	Взлет коптера по сигналу светофора.	2
6	Автономное зависание на высоте не менее 30 см (3 секунды). Касание поля - упражнение не засчитывается.	3
7	**Автономный перелет от т. А до т. Б на высоте не менее 30 см. Штрафы: • Касание стен полигона - 0,5 балла • Касание потолка полигона – 1 балл	4
8	**Автономное зависание над т. Б на высоте не менее 30 см.	5
9	**Автономный сброс груза в т. Б: ▪ малый круг - 6 баллов ▪ средний круг – 4 балла ▪ большой круг – 2 балла	6
10	**Автономный перелет от т. Б до т. В на высоте не менее 30 см. Штрафы: • Касание стен полигона – 1 балл • Касание потолка полигона – 2 балла	5
11	**Автономное зависание над т. В на высоте не менее 30 см (3 секунды).	6
12	Автономная посадка в т. В. Штрафы: • Посадка с вылетом из габаритов метки на 5 см -2 балла • Посадка с вылетом из габаритов метки на 20 см - 4 балла	7
	Всего баллов:	50

Штрафные баллы:

- Высота полета менее 30 см - 0.3 балла
- **Выполнение полета без груза – 50% от полученных баллов
- За использование второй и последующих попыток – Ш=К-0,5×(N-1), где Ш – штрафные баллы, К –количество баллов за данное задание, N – номер попытки

ПРИЛОЖЕНИЕ А

Вариант сборки устройства:

Полное решение:

```
“NTI_Task_AERO.ino”
/*
 Размер системы координат
 max x = 150 см
```

```
 max y = 270 см
 maz z = 140 см
*/
#include <ros.h>
#include <clever/GetTelemetry.h>
#include <clever/Navigate.h>
#include <mavros_msgs/SetMode.h>
typedef ros::NodeHandle_<ArduinoHardware, 3, 3, 100, 100> NodeHandle;
using namespace clever;
using namespace mavros_msgs;
NodeHandle nh;
ros::ServiceClient<Navigate::Request, Navigate::Response> navigate("/navigate");
ros::ServiceClient<SetMode::Request, SetMode::Response>
setMode("/mavros/set_mode");
ros::ServiceClient<GetTelemetry::Request, GetTelemetry::Response>
getTelemetry("/get_telemetry");
////////////////////////////////////////
unsigned long int lastTime;
#define landingDelayLED 5000
void setup() {
  setupIR();
  setupMagnet();
  setupLED();
  // Инициализация rosserial
  nh.initNode();
  // Инициализация сервисов
  nh.serviceClient(navigate);
  nh.serviceClient(setMode);
  nh.serviceClient(getTelemetry);
  // Ожидание подключение к Raspberry Pi
  while (!nh.connected()) nh.spinOnce();
  nh.loginfo("Startup complete");
  // Пользовательская настройка
  // <...>
  // Тестовая программа
  Navigate::Request nav_req;
  Navigate::Response nav_res;
  SetMode::Request sm_req;
  SetMode::Response sm_res;
  GetTelemetry::Request gt_req;
  GetTelemetry::Response gt_res;
  // старт
  loopIR();
  // взлет
  nh.loginfo("Take off");
  nav_req.auto_arm = true;
  nav_req.x = 0;
  nav_req.y = 0;
  nav_req.z = 0.7;
  nav_req.frame_id = "fcu_horiz";
  nav_req.speed = 0.5;
  navigate.call(nav_req, nav_res);
  takeoffLED();
  delaySTOP(3000);
  ();
  delaySTOP(4000);
  nh.loginfo("Fly on point");          // полет в Б // относ карты
  nav_req.x = 0.55;
  nav_req.y = 0.55;
  nav_req.z = 0.80;                    // высота от пола - 70 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = true;
  nav_req.speed = 0.5;
  nav_req.yaw = 1.57;
```

```
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(4000);
  hoveringLED();
  delaySTOP(1000);
  nh.loginfo("Fly on point");          // снижение в Б // относ карты
  nav_req.x = 0.57;
  nav_req.y = 0.54;
  nav_req.z = 1.33;                     // высота от пола - 7 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = true;
  nav_req.speed = 0.4;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(2000);
  hoveringLED();
  delaySTOP(1000);
  nh.loginfo("Fly on point");          // снижение 2 в Б // относ карты
  nav_req.x = 0.58;
  nav_req.y = 0.53;
  nav_req.z = 1.51;                     // высота от пола - 0 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = true;
  nav_req.speed = 0.4;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(1000);
  gt_req.frame_id = "aruco_map"; // фрейм для значений x, y, z
  getTelemetry.call(gt_req, gt_res);
  while (!((gt_res.x >= 0.55 - 0.02) and (gt_res.x <= 0.55 + 0.02) and (gt_res.y
>= 0.55 - 0.02) and (gt_res.y <= 0.55 + 0.02))) {
    gt_req.frame_id = "aruco_map"; // фрейм для значений x, y, z
    getTelemetry.call(gt_req, gt_res);
    nh.spinOnce();
  }
  captureMagnet(0);
  NOcaptureLED();
  // сброс груза
  delaySTOP(2000);
  nh.loginfo("Fly on point");          // поднятие в Б // относ карты
  nav_req.x = 0.57;
  nav_req.y = 0.55;
  nav_req.z = 0.9;                      // высота от пола - 7 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = true;
  nav_req.speed = 0.4;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(2000);
  hoveringLED();
  delaySTOP(1000);
  nh.loginfo("Fly on point");           // полет в В
  nav_req.x = 0.90;
  nav_req.y = 1.51;
  nav_req.z = 0.9;                      //высота от пола - 50 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = false;
  nav_req.speed = 0.5;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
```

```
  delaySTOP(4000);
  hoveringLED();
  delaySTOP(2000);
  nh.loginfo("Fly on point");          // преземление в B
  nav_req.x = 0.90;
  nav_req.y = 1.51;
  nav_req.z = 1.1;                     //высота от пола - 30 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = false;
  nav_req.speed = 0.5;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(2000);
  nh.loginfo("Fly on point");          // преземление 2 в B
  nav_req.x = 0.90;
  nav_req.y = 1.51;
  nav_req.z = 1.47;                     //высота от пола - 30 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = false;
  nav_req.speed = 0.5;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  delaySTOP(2000);
  gt_req.frame_id = "aruco_map"; // фрейм для значений x, y, z
  getTelemetry.call(gt_req, gt_res);
  while (!((gt_res.x >= 0.90 - 0.02) and (gt_res.x <= 0.90 + 0.02) and (gt_res.y
>= 1.51 - 0.02) and (gt_res.y <= 1.51 + 0.02))) {
    gt_req.frame_id = "aruco_map"; // фрейм для значений x, y, z
    getTelemetry.call(gt_req, gt_res);
    nh.spinOnce();
  }
  nh.loginfo("Fly on point");          // преземление 2 в B
  nav_req.x = 0.90;
  nav_req.y = 1.51;
  nav_req.z = 1.54;                     //высота от пола - 30 см
  nav_req.frame_id = "aruco_map";
  nav_req.update_frame = false;
  nav_req.speed = 0.5;
  nav_req.yaw = 1.57;
  navigate.call(nav_req, nav_res);
  flightLED();
  // Посадка
  nh.loginfo("Land");
  sm_req.custom_mode = "AUTO.LAND";
  setMode.call(sm_req, sm_res);
  landingLED();
}
void loop() {
}
void delaySTOP(int vbn) {
  lastTime = millis();
  while (lastTime + vbn > millis()) {
    delay(100);
    nh.spinOnce();
  }
}
“IR_receiver.ino”
#include <IRLibDecodeBase.h>
#include <IRLib_P02_Sony.h>
#include <IRLibCombo.h>
IRdecode myDecoder;
#include <IRLibRecv.h>
```

```
IRrecv myReceiver(9);  //Пин куда подключен сигнал приемника
byte k;
void setupIR() {
  delay(500);
  myReceiver.enableIRIn(); // Запуск ресивера
}
void loopIR() {
  unsigned long int TIMESV = millis();
  bool valueIR = 0;
  delay(2000);
  waitLED();
  while (valueIR == 0) {
//    if (TIMESV + 500 < millis()) {
//      k++;
//      waitLED();
//      TIMESV = millis();
//      if (k == 2) k = 0;
//    }
    if (myReceiver.getResults())
    {
      myDecoder.decode(); // Декодируем...
      if (myDecoder.protocolNum == SONY)  {  }
      myReceiver.enableIRIn();   // Перезапускаем приемник
      if (myDecoder.value == 0x10) {
        captureLED();
        captureMagnet(1);
        valueIR = 1;
      }
    }
    nh.spinOnce();
  }
  delay(500);
}
“LED_strip.ino”
#include "Adafruit_NeoPixel.h"
#define LED_COUNT 30
#define LED_PIN 10
Adafruit_NeoPixel
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
void setupLED() {
  strip.begin();
  NOcaptureLED();
}
void loopLED()
{
}
void captureLED() {
  for (int i = 0; i < LED_COUNT - 1; i++)
  {
    strip.setPixelColor(i, strip.Color(0, 255, 0));
  }
  strip.show();
}
void NOcaptureLED() {
  for (int i = 0; i < LED_COUNT - 1; i++)
  {
    strip.setPixelColor(i, strip.Color(255, 0, 0));
  }
  strip.show();
}
void hoveringLED() {
  for (int i = 0; i < LED_COUNT - 1; i++)
  {
    strip.setPixelColor(i, strip.Color(127, 127, 127));
```

```
  }
  strip.show();
}
void flightLED() {
  for (int i = 0; i < LED_COUNT - 1; i++)
  {
    strip.setPixelColor(i, strip.Color(255, 0, 255));
  }
  strip.show();
}
void takeoffLED() {
  static byte c[5] = {255, 0,  0, 255, 0};
  for (int i = 0; i < LED_COUNT - 1; i++) {
    byte j = i % 3;
    strip.setPixelColor(i, strip.Color(c[j], c[j + 1], c[j + 2]));
  }
  strip.show();
}
void landingLED() {
  lastTime = millis();
  while (lastTime + 5000 > millis() ) {
    for (byte k = 0; k < 3; k++) {
      static byte c[7] = {255, 0,  0, 255, 0, 0, 255};
      for (byte i = 0; i < LED_COUNT - 1; i++) {
        int j = i % 3;
        strip.setPixelColor(i, strip.Color(c[j + k], c[j + k + 1], c[j + k +
2]));
      }
      strip.show();
    }
  }
}
void waitLED() {
  static int c[7] = {200, 200,  0, 200, 200, 0, 200};
  for (int i = 0; i < LED_COUNT - 1; i++) {
    int j = i % 3;
    strip.setPixelColor(i, strip.Color(c[j + k], c[j + k + 1], c[j + k + 2]));
    strip.show();
  }
}
“MagnetCapture.ino”
int pinCapture = 5;
void setupMagnet() {
  pinMode(pinCapture, OUTPUT);
}
void captureMagnet(bool i){
  digitalWrite(pinCapture, i);
}
```

Общие задачи полигона АТС

Задание

Основная задача трека - перемещение в автономном режиме груза через 3 зоны трека: водную, автомобильную и воздушную. Полная задача прохождения трека АТС приносит команде дополнительные баллы. Решением является объединение проездов в рамках всех трех потреков.

Схема полигона

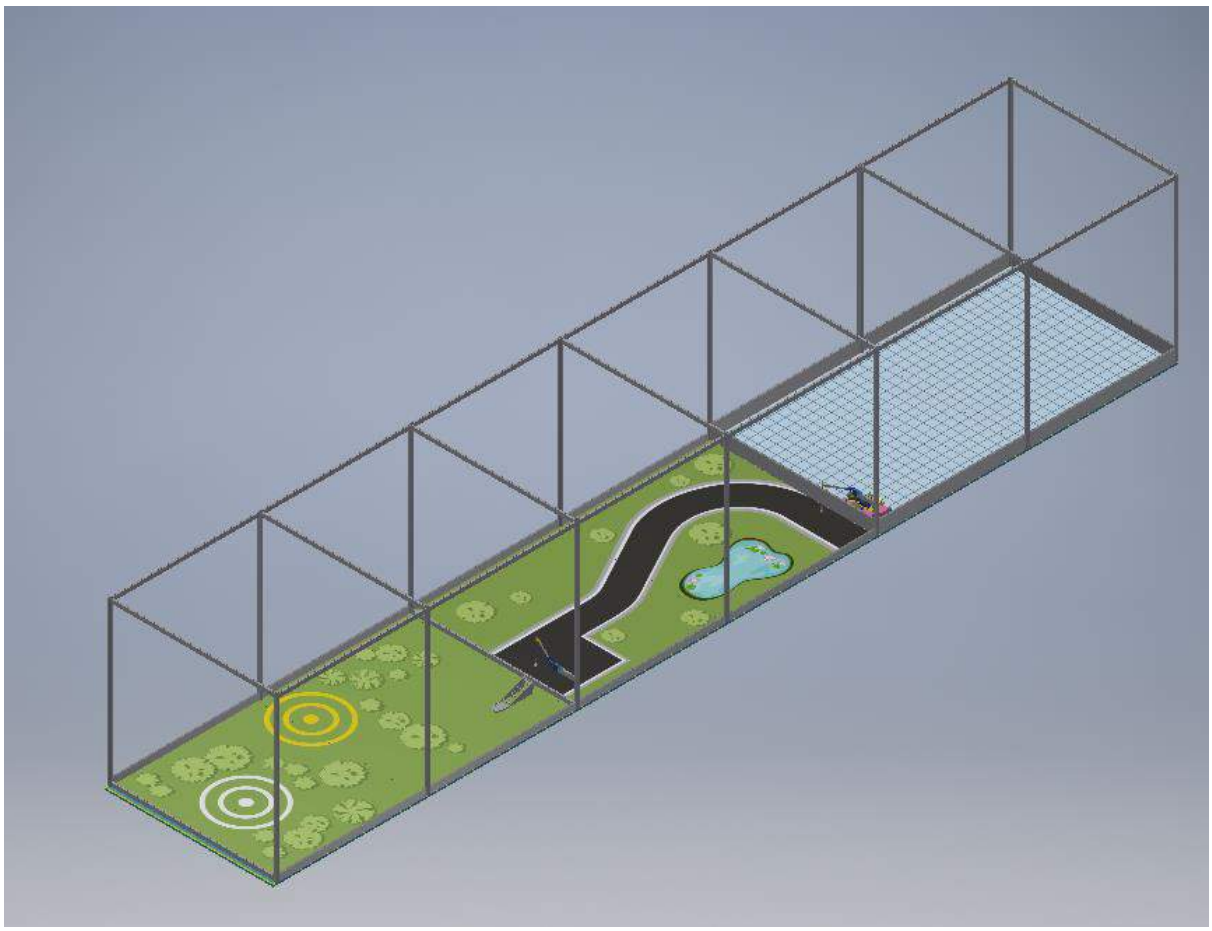


Фото полигона:

Баллы

Перевезти груз по водному, автомобильному и воздушному трекам. Если не указано иначе, применяются штрафы согласно условиям треков.

№	Задача	Баллы
1	Корабль доплыл до нужного порта, остановился и груз перегружен на машину, машина отъехала от зоны парковки	10
2	Машина доехала до зоны коптера и груз перегружен с машинки и захвачен краном	15
3	Груз захвачен коптером	5
4	Коптер доставил груз	20
	Всего баллов:	50

Штрафы:

- Старт транспортного средства произведен не по сигналу маяка – 5 баллов
- Неправильный путь лодки - 5 баллов
- Груз после сброса не попал в большой круг – 5 баллов
- Груз не удалось перегрузить с помощью разгрузочного крана – 5 баллов
- Груз сброшен вне круга в зоне коптеров – 5 баллов
- Касание стенок полигона и его элементов – 1 балл на задачу

Общие правила

Для всех задач используются общие дополнительные правила работы на полигоне.

Требования:

- Все модели начинают выполнение (сдачу) задания после нажатия кнопки старт на пульте управления (Включается светофор на полигоне). Кнопку нажимает преподаватель, если в задании не указано иное.
- Модель не должна рассыпаться в руках (как критерий: выдерживать падение с высоты 5 см на стол полигона).

- Указания светофоров и маяков разыгрываются жеребьевкой перед стартом прохождения трека.
- Контроль питания - зона ответственности команды. Разряженная батарея во время сдачи попытки не является уважительной причиной.

Штрафы

№	Штрафы	Баллы
1	Разрядка аккумулятора во время сдачи задачи включая спутник	Списание попытки / зачет как есть
2	Запуск коптера вне полигона и без преподавателя	10 баллов
3	Заход в зону преподавателей без разрешения преподавателя	5 баллов
4	Устройство не начало выполнение задания после нажатия преподавателем кнопки старта	Списание попытки
5	За использование второй и последующих попыток внутри задачи подтрека	$Ш = К - 0{,}5 \times (N - 1)$, где Ш – штрафные баллы, К –количество баллов за данное задание, N – номер попытки