





## Оглавление

Введение.....	4
Как решается проблема с парковками в мире и у нас в стране.....	5
Теоретическое обоснование решения.....	7
Определение.....	8
Требования к установке систем .....	8
Описание проекта .....	8
Электроника.....	9
Программный — локальный.....	9
Программный — серверный.....	10
Общее приписание.....	12
Литература и электронные ресурсы:.....	14
Приложение.....	15
Код микроконтроллера:.....	15
Телефон(вместе с заготовками под умный город):.....	20
Сервер:.....	29

## Введение

Согласно результатам исследования, проведенного аналитическим агентством «АВТОСТАТ», по состоянию на 1 июля 2017 года обеспеченность легковыми автомобилями в среднем по России составляла 290 штук на 1 тыс. жителей. По состоянию на 1 июля 2018 года в Новосибирской области насчитывалось 799,5 тысячи легковых машин. В рейтинге по количеству автомобилей Новосибирская область заняла 14-е место из 20 возможных. На тысячу человек в городе приходится 273 авто. Решение проблемы хранения автомобилей в центре города власти Новосибирска видят в ограничении притока автомобилей, в том числе за счет организации платных парковок. Первая такая парковка начала работу в октябре.



Сегодня машина есть у каждого четвёртого новосибирца. При этом эксперты считают, что введение регуляторов необходимо уже в тех ситуациях, когда на тысячу человек – 200 авто. Тем более что все эти автомобили где-то нужно хранить, и не только ночью, но и днем. Самое простое решение – оставить машину на правой полосе улицы рядом с офисом. Именно так и поступало подавляющее большинство автовладельцев, а центральные улицы города все сильнее напоминали парковку. По решению мэрии уже летом 2013 года началась ликвидация парковочных карманов на Красном проспекте. Под удар попали сразу 46 из них, расположенные от пересечения с ул. Ядринцевской до ул. Писарева. Удобства для парковки автотранспорта это явно не добавило. Каждая поездка автомобилиста начинается с места парковки в пункте «А» и заканчивается парковкой в пункте «Б». По действующему

СНиПу минимальным размером парковочного места является 2,5 на 5,3 метра.

Количество автомобилей растет, при этом водители в среднем используют от двух до пяти стоянок в день.

Применяя обычную логику, несложно догадаться, что проблема размещения автотранспорта в больших городах становится всё более острой.

### **Как решается проблема с парковками в мире и у нас в стране**



Кто не видел фотографий из Токио или Нью-Йорка, где на парковочных «пяточках» стоят стеллажи из машин? А ажурные двадцатипятиэтажные стеклянные башни-гаражи фольксвагеновского комплекса Autostadt в Вольфсбурге - с лифтами, которые автоматически расставляют новенькие машины штабелями? Есть ли нечто подобное у нас в стране? Парковки-

этажерки... Можно ли с их помощью решить парковочную проблему хотя бы у себя во дворе?



Девятиэтажную полностью автоматическую парковку на 50 машин в Останкине построили еще в 2009 году, но из-за волокиты официально открыли только в феврале 2013 года



Проблемы - дороговизна и бумажная волокита. Цена машино-места в многоярусном механизированном паркинге начинается примерно с 700 тысяч рублей, и даже в легких модульных конструкциях цену не удастся опустить ниже 200-300 тысяч рублей. Вдобавок согласование нужных документов может занимать несколько лет, срок окупаемости растягивается, а издержки от общения с чиновниками сводят на нет всю экономическую выгоду.



Две автоматизированные парковки установлены в Люберцах — одна на 21, другая на 24 автомобиля. Причем эти комплексы закладывались сразу при строительстве нового квартала

Районные власти, может, и рады были бы оптимизировать стоянки на своей территории, но ни средств, ни нужной законодательной базы и понятных технических нормативов для этого не имеют.

А раз так - может, взять инициативу в свои собственные руки и, скинувшись с соседями, установить «этажерку» у себя во дворе?

Простейший вариант - лифт-подъемник наподобие тех, что используются в автосервисах. Первый автомобиль просто поднимается над землей примерно на два метра - так, чтобы под ним разместилась вторая машина: цена вопроса —

около 80 тысяч рублей за машино-место.



Модульный двухъярусный мини-паркинг с независимой погрузкой по принципу «пятнашек» — пять

машино-мест и одна свободная ячейка. (Во дворе одного из домов на Воронцовской улице )

В многоместной модульной системе, где машины стоят на подвижных палетах и могут въезжать-выезжать независимо друг от друга, машино-место дороже - рассчитывать стоит на сумму минимум 180 тысяч рублей. Теоретически можно поставить во дворе и напоминающую карусель роторную парковку на десяток машин, но стартовая цена каждого машино-места в этом случае - от 360 тысяч рублей.

Однако сюда нужно прибавить еще затраты и хлопоты на проведение электропитания и монтаж. Но главное - оформление прав на клочок «парковочной» земли.

Далее - электропитание: электроприводы, как правило, требуют трехфазной сети на 380 вольт. Причем обычному двухместному лифту-подъемнику нужна мощность около 1,5 кВт, а многоместной «этажерке» - уже от 8 до 18 кВт.

Плюс защитное укрытие, которого требуют некоторые системы, плюс регулярное техобслуживание механизмов, плюс затраты на электроэнергию - все это тоже ляжет дополнительными денежным бременем.



Вертикальный 15-ярусный «пенал» в Москве на улице Климашкина — редкий пример автоматизированной парковки, пристроенной непосредственно к жилому дому. На клочке в 127 м<sup>2</sup>

Часть проблемы снимают подземные парковки, закладываемые на начале строительства дома. К сожалению, реалии таковы, что большинство домов нашего города не имеют таких парковок, и большинству автовладельцев требуется большая выдержка и виртуозность, чтобы припарковать свой автомобиль хотя бы на

ночь у дома. К вечеру найти парковочное место крайне затруднительно. Как быть?

Выход есть и в его основе лежат открытые парковки на основе беспроводных технологий.

## **Теоретическое обоснование решения**

## **Определение**

**Умная парковка (smart parking)** – специализированное место для парковки автомобилей, созданное с использованием датчиков и современных технологий для быстрого и удобного поиска парковочных мест, обеспечения безопасности и автоматизации процесса постановки автомобиля на стоянку. Системы данного типа состоят из сетей беспроводных датчиков, которые обнаруживают в режиме реального времени, занято парковочное место или нет. Точная информация о состоянии парковки может быть эффективна и полезна для водителей.

## **Требования к установке систем**

Городские улицы не должны быть надолго закрыты или недоступны в течение многих дней на время установки сенсорной сети, так как это будет иметь огромное влияние на поток движения в городе. Таким образом, любая система, которая может быть использована в городе, требует, чтобы она была простой в установке и настройке, хорошо масштабируемой и надежной.

## **Описание проекта**

Основной задачей проекта является внедрение программно-аппаратного комплекса, позволяющего в режиме реального времени отслеживать состояние каждого парковочного места. В мире уже есть наработки, но в большинстве своём они заграничные, хоть в России есть очень интересные проекты, но они мало развиты и некоторые работают по определению местоположения всех машин в городе, что с одной стороны позволяет видеть всю ситуацию, а с другой GPS имеет большую погрешность. Я же предлагаю внедрять в каждое парковочное место датчик, который определяет наличие или отсутствие транспортного средства. Далее микроконтроллер Ардуино, обрабатывает показания, изменяет состояние в своей памяти каждого места, устанавливает индикацию, светодиод горит: зелёный — свободно, красный — занято, жёлтый и также поворачивается заслонка — забронировано; передаёт информацию состояния на сервер, чтобы каждый желающий мог скачав приложение посмотреть занятые/свободные места и произвести бронирование. Сам контроллер Arduino будет иметь выход в глобальную сеть с помощью wifi или ethernet модулей, осуществляя связь с сервером. В дальнейшем микроконтроллер будет отправлять информацию в формате xml о карте парковки. К серверу подключаются клиенты — смартфоны, которые имеют возможность просматривать состояние всех мест и бронировать



свободные. Если происходят изменения на парковки, они транслируются через сервер всем подключённым устройствам.

Система разделяется на следующие уровни:

- 1) электроника
- 2) программный — локальный
- 3) программный — серверный

### **Электроника**

Выбор датчиков, модулей; разработка схем.

Были рассмотрены следующие варианты датчиков:

- 1) ультразвуковой датчик расстояния
- 2) инфракрасный датчик расстояния
- 3) пирозлектрический датчик движения
- 4) камера
- 5) ёмкостный датчик
- 6) магнитный датчик

Первые два мало эффективны: они поглощаются легко мягким материалом и искажаются. Третий чувствительный(может «обратить своё внимание» на человека) и реагирует на движение. Четвёртый дорог, и грязь, растительность, время суток или иные факторы влияют на видимость. Пятый по своей природе очень эффективный, но готовые датчики на рынке бывают дорогие и очень чувствительны. Шестые дороги и могут быть подвержены влиянию окружающих магнитных полей.

Несмотря что все имеют минусы, технология ёмкостного датчика является наиболее эффективной. Сама технология основывается на изменении ёмкости датчика под взаимодействием с окружающими объектами. А самим датчиком может быть железная входная дверь, поэтому допустим использование прикрученных к асфальту металлических пластин не доставит проблем с финансами, установкой и эффективностью. Но на макете будет использован ультразвуковой датчик, т. к. пока важной задачей стоит разработки системы и логики всей сети.

Для управлением заслонкой возможно использование сервомотора или шагового мотора в связке в микрокнопками.

Для доступа в глобальную сеть интернет будет использован модуль wifi esp8266.

### **Программный — локальный**

Программа для Arduino и телефона.

Программа для микроконтроллера Arduino Mega писалась в Arduino IDE на си-подобном языке. Микроконтроллер должен проверять заняты ли какие-либо места, поддерживать бронирование и отправлять состояние на сервер. Также над каждым местом реализована индикация: красный — занята, зелёный — свободно, жёлтый — забронировано, и, если место забронировано, поворачивается заслонка с надписью «БРОНЬ».

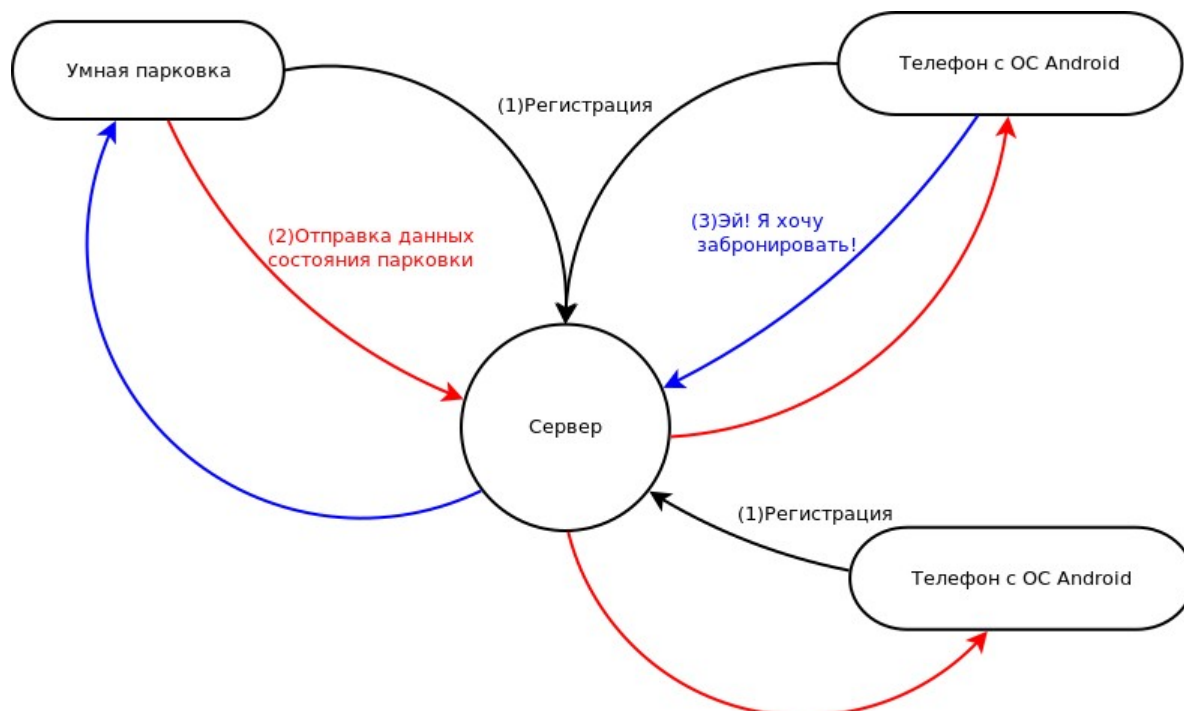
#### *ПО для Android*

Приложение для телефона реализует графический интерфейс для просмотра места, бронирования и разбронирования. Место серое — свободно, красное — занято, жёлтое — забронировано. Использовалась среда Processing IDE, имеющая удобную библиотеку для сетевых подключений и передачи данных через интернет. Самой парковкой реализовано если в течении 15мин вы не приедете ваше место может занять кто-либо другой.

#### **Программный — серверный**

Сервер был написан самостоятельно и больше для системы умного города, в которую входит и парковка, поэтому для облегчения совместимости выбран этот путь, на языке программирования Java в среде программирования Processing IDE, она имеет очень удобные библиотеки для работы с клиентами и серверами и изначально я умел программировать в ней.

Рассмотрим схему примера связи клиентов и парковки:



Чтобы общаться с другими объектами надо пройти «регистрацию», для этого при подключении надо отправить:

SYSSC: HELLO

ID1: identifier

identifier — это числовой код группы — идентификатор. Устройства принадлежащие к одной группе могут спокойно общаться.

На схеме видно что при регистрации каждое устройство общается только с сервером, а после парковка одним сообщением может отправлять всем клиентам состояние мест.

К программе для Arduino — прибавляется код соединения с сервером, регистрацией и обработкой локальных парковочных команд.

К программе для телефона — подключение к серверу, регистрация, отправка команд и приём ответов.

Как только происходят изменения на парковке, Arduino отправляет - !  
getpark!012210\n

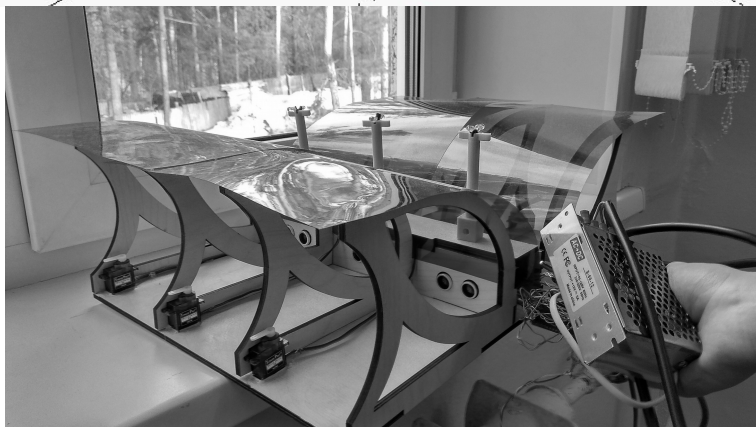
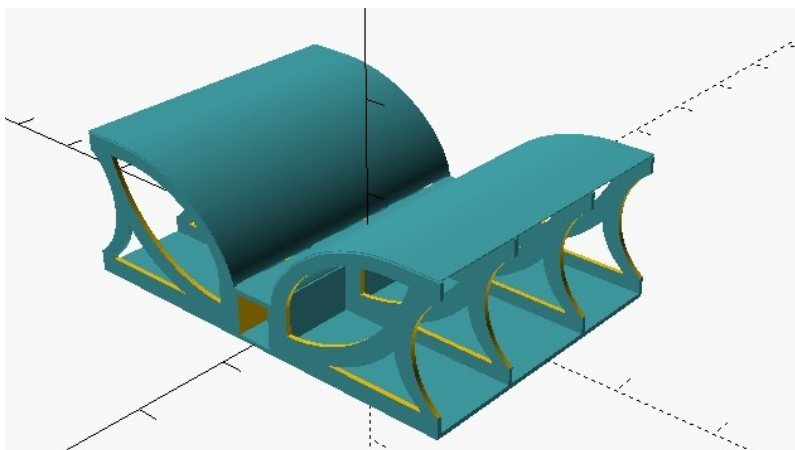
где 012210 — это состояние парковочных мест.

Если клиент бронирует, он отправляет — setpark 0\n

где 0 — идентификатор места == индекс в массиве мест.

### **Макет**

Для реализации проекта был разработан дизайн парковки, предусматривающий защиту парковки от непогоды. Макет был смоделирован в программе САПР OpenSCAD, чертежи сделаны в QCAD Community Edition, и детали макета вырезаны лазером с системой ЧПУ из фанеры 5мм.



## **Общее приписание**

Полученная схема проста в реализации и её просто можно устанавливать не только в торговых центрах, но и в районах. В дальнейших планах развития проекта: изучение ёмкостных датчиков, улучшение, исправление неполадок в системе, дальнейшая проработка ПО для Android, детальный разбор проблемы установки «куда хочу», детальный просчёт цены и внедрение в договорённости с ЖКХ и администрацией в Ленинский район.

Также существуют, пока неизвестные, проблемы, о которых мы не знаем, но решение этой задачи ложится на плечи тестировщиков ПО.

Также мы предлагаем несколько вариантов комплектации+цена(без учёта проводки, стоек, только электронные компоненты):

### *бюджетный*

в состав только индикация и определение

«мозг» - 1000р

цена места - ~400р

Итого на 6 мест: ~3500р

### *стандарт*

в состав прибавляется сервомоторы для бронирования

цена места: ~1000р

Итого на 6мест: 7000р

## **Заключение**

Для чего я задумал этот проект? На самом деле этот проект – продолжение проекта «Умный дом как прототип умного города», над которым я с друзьями работаем уже второй год. Глядя на улицы и дворы нашего города, трудоемкий процесс парковки машины родителями в конце рабочего дня, я начал понимать глубину проблемы. С ростом числа автомобилей в городе (а ведь есть еще и гости города, транзитные машины, которые тоже как-то где-то временно оставить) эта проблема становится все острее и острее. Конечно, можно ослабить проблему многоярусными парковками, подземными в вырастающих новостройках, а как быть с уже сложившимися обстоятельствами? Я считаю, что мой вариант парковки может быть использован как в новых микрорайонах с выделенными площадками, так и в микрорайонах старой застройки. Я создал модель с двусторонними площадками, но их спокойно можно трансформировать и в односторонние. В моей модели я постарался создать максимально возможные в этой ситуации удобства для автомобилистов, но можно использовать и бюджетный вариант, оставив лишь функцию поиска свободного места и его бронирования. Еще



хотелось бы оснастить каждую такую парковку устройством для подзарядки электромобилей.

### **Литература и электронные ресурсы:**

1. Иго Том Arduino, датчики и сети для связи устройств. [+ исходный код] (2-е издание, 2015)
2. <https://all-arduino.ru/knigi-po-arduino/>
3. Изучаем Arduino. Джереми Блум
4. Проекты с использованием контроллера Arduino. Петин В.А
5. Програмируем Arduino. Саймон Монк
6. Практическая энциклопедия Arduino
7. <https://processing.org/>
8. <https://android.processing.org/>
9. <https://www.qcad.org/en/>
10. <http://www.openscad.org/>
11. <http://www.openscad.org/cheatsheet/index.html>
12. <https://www.arduino.cc/>
- 13.** Processing: A Programming Handbook for Visual Designers, Second Edition. Casey Reas and Ben Fry.
14. Make: Getting Started with Processing, Second Edition. Casey Reas and Ben Fry.

## Приложение

### Код микроконтроллера:

```
////////////////////////////////////
//класс для места на парковке, выполняет обновление состояния //
//места, индикации и сервомоторов - логика каждого места //
//конструктор отсутствует, для инициализации используется метод//
//begin(trigpin, echopin, redledpin, greenledpin, servopin) //
// trigpin, echopin - пины ультразвукового датчика //
// redledpin, greenledpin - пины красного и зелёного //
// светодиодов соответственно //
// servopin - пин для управления сервомотором //
//up() - метод должен вызываться каждый цикл для обновления //
// места //
//zabr(IDforPark) - метод для бронирования места //
// IDforPark - строка идентификатора для бронирования //
//Rz(IDforPark) - метод для разбронирования //
// IDforPark - см zabr(IDforPark). //
////////////////////////////////////

#include <Servo.h>
class Mesto {
public:
    uint8_t ismesto = 0; // состояние места
    uint8_t trig; //пин для создания ультразвукового импульса
    uint8_t echo; // пин для принятия ультразвукового импульса
    uint8_t led1; // пин красного светодиода
    uint8_t led2; // пин зелёного светодиода
    boolean state = false; //состояние светодиода
    unsigned long timerT = 0; //таймер бронирования
    String myid = ""; //ID бронирующего клиента
    int distance = 17; //максимальная дистанция
    int last = 0; //прошедший градус сервомотора
    Servo s;
    void begin(uint8_t _trig, uint8_t _echo, uint8_t _led1, uint8_t _led2,
uint8_t _s) { // функция для инициализации места
        trig = _trig;
        echo = _echo;
        led1 = _led1;
        led2 = _led2;
        pinMode(trig, OUTPUT);
        pinMode(echo, INPUT);
        pinMode(led1, OUTPUT);
        pinMode(led2, OUTPUT);
        s.attach(_s);
        s.write(45);
    }
    void zabr(String now_id) { // функция для бронирования; аргумент - строка
- ID брони
        if (myid.equals("")) {
            ismesto = 1;
            myid = now_id;
            timerT = millis();
        }
    }
}
```

```

void Rz(String now_id) { // разфункция для бронирования; аргумент -
строка - ID брони
  if (!myid.equals("")) {
    if (now_id.equals(myid)) {
      ismesto = 0;
      myid = "";
    }
  }
}

void up() { // функция для обновления места(есть ли машина, какой цвет
индикации, поворот заслонки и проверка выхода таймаута)
  int cm;
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(5);
  digitalWrite(trig, LOW);
  cm = pulseIn(echo, HIGH, distance * 2 * 29) / 29 / 2;
  if ((cm != 0) && (cm < 16)) {
    ismesto = 2;
  } else if (ismesto != 1) {
    ismesto = 0;
  }
  switch (ismesto) {
    case 2:
      state = false;
      if (last != 45) {
        s.write(45);
        last = 45;
      }
      break;
    case 1:
      digitalWrite(led1, HIGH);
      digitalWrite(led2, HIGH);
      if (last != 135) {
        s.write(135);
        last = 135;
      }
      if (millis() - timerT > 15 * 1000) {
        ismesto = 0;
        myid = "";
      }
      Serial.println("in zabr");
      break;
    case 0:
      state = true;
      ismesto = 0;
      myid = "";
      if (last != 45) {
        s.write(45);
        last = 45;
      }
      break;
    default:
      break;
  }
  if (ismesto != 1) {

```





```

void loop() {
  //если прошла секунда обновляем места и отправляем состояние на сервер
  if (millis() - t1 > 1000) {
    for (int i = 0; i < 6; i++) {
      ms[i].up();
      parkstate[i] = char(ms[i].ismesto + 48);
    }
    esp.print("!getpark!" + String(parkstate) + "\n");
    t1 = millis();
  }
  //m.ismesto set or get of 0 - свободно, 1 - забронированию, 2 - занято

  //если есть данные то считываем до '\n' и проверяем на команда, если так то
  выполняем
  if (esp.available() > 0) {
    String msg = esp.readStringUntil('\n');
    Serial.println(msg);
    if (msg.lastIndexOf("sp") > -1) {
      int i = msg.substring(6, 7).toInt();
      String IDf = msg.substring(7, msg.length());
      ms[i - 1].zabr(IDf);
    } else if (msg.lastIndexOf("it") > -1) {
      int i = msg.substring(6, 7).toInt();
      String IDf = msg.substring(7, msg.length());
      ms[i - 1].Rz(IDf);
    }
  }
}

//если прошла минута проверяем соединение
if (millis() - t > 60*1000) {
  esp.flush();
  esp.print("?????\n");
  String tmp = "";
  t = millis();
  while (true) {
    if (millis() - t > 2000) {
      reconnect();
      break;
    }
    if (esp.available() > 0) {
      tmp += String(char(esp.read()));
    }
    if (tmp.indexOf("!") > -1) {
      break;
    }
  }
  t = millis();
}

}

////////////////////////////////////
//function for connect to server    /
//ROSTELEKOM - ssid for wifi        /
//lenovozuk123 - password for wifi  /
//192.168.1.3 - ip server            /
//48999 - port for server            /
//ID1: -1 - id for init in server   /

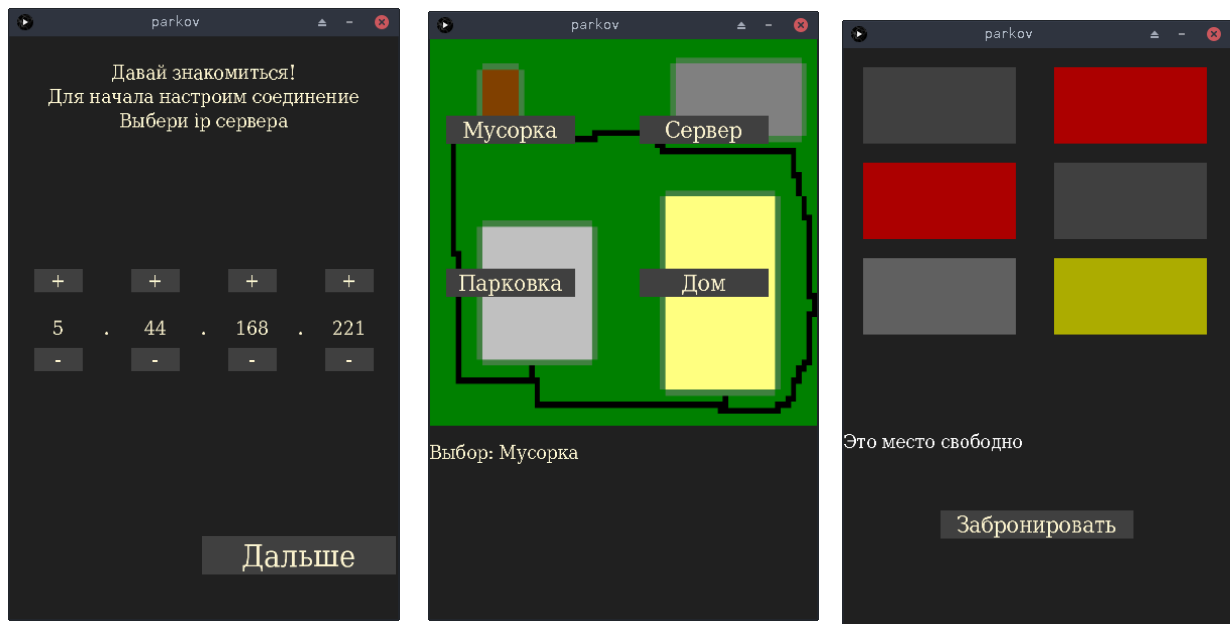
```

```

////////////////////////////////////
void reconnect() {
    esp.print("+++");
    delay(500);
    if (!sendcmd("AT", "OK", 250))return;
    // if (!sendcmd("AT+CWMODE=1", "OK", 250))return;
    if (!sendcmd("AT+CWJAP=\"Kelf\", \"rebekka5\"", "OK", 7000))return;
    //if (!sendcmd("AT+CIPSTART=\"TCP\", \"192.168.1.3\", 48999", "OK",
5000))return; //94.180.117.139
    esp.print("AT+CIPSTART=\"TCP\", \"192.168.43.71\", 48999\r\n");
    delay(1000);
    if (!sendcmd("AT+CIPMODE=1", "OK", 250))return;
    esp.print("AT+CIPSEND\r\n");
    delay(1000);
    esp.print("SYSSC: HELLO\nID1: -1\n");
}
////////////////////////////////////
//function for send command to esp///
//ARGUMENTS(cmd, ans, timeout):      /
//cmd - String - command (example:AT)/
//ans - String - true answer         /
//      to command (example:OK)      /
//timeout - int - timeout             /
//      for wait answer               /
//                                   /
//return true:                        /
// if esp return answer in timeout    /
//return false:                       /
// if esp not return answer           /
// in timeout                         /
////////////////////////////////////
boolean sendcmd(String cmd, String ans, int timout) {
    String tmp = "";
    unsigned long t = millis();
    int count = 0;
    esp.print(cmd + "\r\n");
    while (tmp.indexOf(ans) < 0) {
        if (esp.available() > 0) {
            tmp += (char)esp.read();
        }
        if (millis() - t > timout) {
            esp.print(cmd + "\r\n");
            count++;
            if (count == 15) {
                return false;
            }
        }
        t = millis();
    }
    return true;
}

```

## Телефон(вместе с заготовками под умный город):



```
import processing.core.*;
import processing.data.*;
import processing.event.*;
import processing.opengl.*;

import java.io.*;
import java.lang.reflect.*;
import java.net.*;

import java.util.HashMap;
import java.util.ArrayList;
import java.io.File;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;

public class parkov extends PApplet {

    public Client c; //клиент подключения к серверу
    int dw, dh;      //размеры экрана
    ArrayList hellob=new ArrayList(); //массив кнопок для приветственного окна
    int bc1 = color(64);           //
    int bc2 = color(96);           //цветовая схема
    int bc4 = color(0xFE, 0xF5, 0xD0); //с помощью инструментов adobe
    int bc3 = color(128);          //
    int state=0;                  //состояние программы
    PFont myfont;                 //шрифт
    int textsize;                 //размершрифта
    String myId ="13";            //ID пользователя в буд. получается из базы данных
    JSONObject data;              //используется для хранения данных приложения
```



```

public void settings() {           //настройка размера окна(только для режима
ПК)
    float k=0.8f;
    size(PApplet.parseInt(480*k), PApplet.parseInt(720*k));
}

public void setup() {              //установки
    orientation(PORTRAIT);         //ориентация портретная
    background(32);                //цвет фона
    dw=width;                      //получение размеров экрана
    dh=height;
    myfont=loadFont("DejaVuSerifCondensed-64.vlw"); //загрузка шрифта
    data = loadJSONObject("./data/data.json");      //загрузка данных
    textFont(myfont);              //установка шрифта
    textsize=dh/30;                //расчёт размера текста
    fill(255);                     //заливка
    textSize(textsize);            //установка размера текста
    inithello();                   //инициализация окон
    initcard();
    initparkovka();
    initmus();
    inithome();
}

public void draw() { //основной цикл
    background(32); //обновление кадра
    switch(state) { //определение состояния ==> логика сейчас
    case -2:
        delay(6000);
        exit();
        break;
    case -1: //в случае ошибки
        textAlign(CENTER);
        text("\nИзвините,\nчто то пошло не так...\n(1)проверьте интернет
соединение\n(2)проверьте правильность ip\n(3)попробуйте позже\ndo
встречи...", dw/2, textsize);
        state = -2;
        break;
    case 0:
        hello();
        break;
    case 1:
        break;
    case 2:
        card();
        break;
    case 3:
        parkovka();
        break;
    case 4:
        mus();
        break;
    case 5:
        home();
        break;
    }
}
}

```

```

public void keyPressed() { //для ввода ip адреса сервера
    if (state==0) {
        if (key>=48 && key<=58) {
            ip[PApplet.parseInt(cursor/3)]=(ip[PApplet.parseInt(cursor/3)]*10+(key-
48));
            ip[PApplet.parseInt(cursor/3)]%=1000;
            cursor++;
            if (cursor==12) {
                cursor=0;
            }
        }
    }
}
public void backPressed() { //если нажата кнопка назад то устанавливаем
главное меню
    state = 2;
    isloop=false;
}

class Button { //класс кнопки писался самостоятельно, не буду расписывать
    int x, y, w, h; //мало времени и не несёт основной логики проекта
    int c1, c2, c3, ct=color(0);
    boolean last=false, now=false, press=false;
    PImage bi;
    String text = "";
    boolean istext=true;
    Button(int xt, int yt, int wt, int ht, int c1t, int c2t, int c3t, String
textt, int ctt) {
        x=xt;
        y=yt;
        w=wt;
        h=ht;
        c1=c1t;
        c2=c2t;
        c3=c3t;
        text = textt;
        ct=ctt;
    }
    Button(int xt, int yt, int wt, int ht, int c1t, int c2t, int c3t, PImage
bit) {
        x=xt;
        y=yt;
        w=wt;
        h=ht;
        c1=c1t;
        c2=c2t;
        c3=c3t;
        istext=false;
        bi=bit;
    }
    public void up() {
        noStroke();
        if (mouseX>x && mouseX<x+w && mouseY > y && mouseY<y+h) {
            if (mousePressed) {
                if (istext)fill(c3);
                else tint(c3);
            }
        }
    }
}

```

```

        } else {
            if (istext)fill(c2);
            else tint(c2);
        }
    } else {
        if (istext)fill(c1);
        else tint(c1);
    }
    if (istext)rect(x, y, w, h);
    else image(bi, x, y, w, h);
    textAlign(CENTER, CENTER);
    fill(ct);
    textSize(h*0.8f);
    text(text, x+w/2, y+h*0.5f);
    now=(mouseX>x && mouseX<x+w && mouseY > y && mouseY<y+h && mousePressed);
    if (now!=last) {
        if (now && !last) {
            last = now;
            press= true;
        }
    }
    last=now;
    return;
}

public boolean press(){
    boolean tmp = press;
    press=false;
    return tmp;
}

};
PImage maps; //изображение карты
String obm[] = {"Мусорка", "Сервер", "Парковка", "Дом"}; //вкладки
ArrayList cardb=new ArrayList(); //кнопки
int cursorv=0; //выбранная вкладка
public void initcard() { //инит. загрузка изображений, и настройка
кнопок
    maps=loadImage("map.png");
    maps.resize(dw, dw);
    Button b;
    b= new Button(dw/24, dw/5, dw/3, dh/20, bc1, bc2, bc3, obm[0], bc4);
    cardb.add(b);
    b= new Button(dw/24*13, dw/5, dw/3, dh/20, bc1, bc2, bc3, obm[1], bc4);
    cardb.add(b);
    b= new Button(dw/24, dw/5*3, dw/3, dh/20, bc1, bc2, bc3, obm[2], bc4);
    cardb.add(b);
    b= new Button(dw/24*13, dw/5*3, dw/3, dh/20, bc1, bc2, bc3, obm[3], bc4);
    cardb.add(b);
}
String msgc = "";//отладочно-информативная строка
public void card() {
    image(maps, 0, 0); //прорисовка карты
    textSize(textsize); //настройка текста
    textAlign(LEFT, TOP);
    text("Выбор: "+obm[cursorv]+"\\n"+msgc, 0, dw+textsize); //отображение
отладочно-информативной строки
    for (int i =0; i<cardb.size(); i++) {//обновление всех кнопок и
произведение соответствующих действий

```

```

Button b = (Button)cardb.get(i);
b.up();
if (b.press()) { //если кнопка нажата то
    if (i==2) {//переходим к парковке
        msgc="";
        state = 3;
        c.stop();
        connect("-1", 3);
        delay(100);
    } else if (i==1) {//это сервер
        msgc="Вы не имеете прав";
    } else if (i==0) {//мусорка
        msgc="";
        c.stop();
        connect("-2", 4);
        delay(100);
    }else if(i==3){//дом
        msgc="";
        c.stop();
        connect("13", 5);
        delay(100);
    }
    } else {
        msgc="Данный раздел пока не поддерживается\nПросим извинения.\nПриложение в разработке";
    }
    cursorv=i;
}
}
//hello
int cursor=0; //разряд ип адреса
int ip[]={5, 44, 168, 221};//ip address

public void inithello() {//инит
    Button b;
    b = new Button(dw/16, dh/5*2, dw/8, dh/25, bc1, bc2, bc3, "+", bc4);
    hellob.add(b);
    b = new Button(dw/16*5, dh/5*2, dw/8, dh/25, bc1, bc2, bc3, "+", bc4);
    hellob.add(b);
    b = new Button(dw/16*9, dh/5*2, dw/8, dh/25, bc1, bc2, bc3, "+", bc4);
    hellob.add(b);
    b = new Button(dw/16*13, dh/5*2, dw/8, dh/25, bc1, bc2, bc3, "+", bc4);
    hellob.add(b);
    b = new Button(dw/16, dh/13*7, dw/8, dh/25, bc1, bc2, bc3, "-", bc4);
    hellob.add(b);
    b = new Button(dw/16*5, dh/13*7, dw/8, dh/25, bc1, bc2, bc3, "-", bc4);
    hellob.add(b);
    b = new Button(dw/16*9, dh/13*7, dw/8, dh/25, bc1, bc2, bc3, "-", bc4);
    hellob.add(b);
    b = new Button(dw/16*13, dh/13*7, dw/8, dh/25, bc1, bc2, bc3, "-", bc4);
    hellob.add(b);
    b = new Button(dw/10*5, dh/15*13, dw/2, dh/15, bc1, bc2, bc3, "Дальше",
bc4);
    hellob.add(b);
}
boolean keyb=false;//переменная для открытия клавиатуры
public void hello() {

```

```

        if (!keyb) { //открытие клави
            //openKeyboard();
            keyb=true;
        }
        textAlign(CENTER); //информационное сообщение
        text("\nДавай знакомиться!\nДля начала настроим соединение\nВыбери ip
сервера", dw/2, textsize);
        for (int i =0; i<hellob.size(); i++) { //обновление кнопок и изменение ip
            Button b = (Button)hellob.get(i);
            b.up();
            if (i==8) {
                if (b.press()) {
                    if (keyb) {
                        //closeKeyboard();
                        keyb=false;
                    }
                    connect(myId, 2);
                }
            } else {
                if (b.text.equals("+")) {
                    if (b.press()) {
                        ip[i]++;
                        if (ip[i]>255)ip[i]=0;
                    }
                } else {
                    if (b.press()) {
                        ip[i-4]--;
                        if (ip[i-4]<0)ip[i-4]=255;
                    }
                }
            }
        }
        textSize(textsize); //отображаем ip
        text(str(ip[0]), dw/8, dh/2);
        text(".", dw/8*2, dh/2);
        text(str(ip[1]), dw/8*3, dh/2);
        text(".", dw/8*4, dh/2);
        text(str(ip[2]), dw/8*5, dh/2);
        text(".", dw/8*6, dh/2);
        text(str(ip[3]), dw/8*7, dh/2);
    }

    public void connect(String group, int nstate) { //функция для соединения с
сервером по нужному ID группы и переходящее окно
        try {
            if (c==null || !c.active() || c.ip() == null) {
                c=new Client(this, str(ip[0])+"."+str(ip[1])+"."+str(ip[2])
+str(ip[3]), 48999);
            }
            if (c==null || !c.active() || c.ip() == null) {
                state = -1;
                return;
            }
            c.write("SYSSC: HELLO\nID1: "+group+"\n");
        }
    }

    catch (Exception ex) {
        state = -1;
        return;
    }
}

```

```

    }
    state=nstate;
}
long t12=0;//в разработке
public void inithome() {

}

public void home() {
    if(millis()-t12>250){
        if(mouseY>dh/10){
            c.write("PI"+str(map(mouseY,dh/10,dh,0,1000))+"\n");
        }else{
            c.write("PI0\n");
        }
        t12=millis();
    }
}

int datamus[]={46,23,95};//пример данных мусорки
public void initmus() {
}

public void mus() {//основной цикл
    if (!isloop)//запуск паралейного потока для работы с сетевыми данными
        thread("musup");
    fill(64); //рисуем столбики заполненности
    noStroke();
    for(int i=0;i<3;i++){
        rect(dw/5*2/4*(i+1)+dw/5*i,dh/10,dw/5,dh/3*2);
    }
    fill(128);
    for(int i=0;i<3;i++){
        rect(dw/5*2/4*(i+1)+dw/5*i,dh/10+map(datamus[i],0,100,dh/3*2,0),dw/
5,dh/3*2-map(datamus[i],0,100,dh/3*2,0));
    }
}

public void musup() {//проверяем на входящие данные, обрабатываем, и
устанавливаем в массив для визуализации
    isloop=true;
    while (true) {
        if (c.available()>0) {
            String msgp=c.readStringUntil('\n');
            if (msgp!=null) {
                if (msgp.indexOf("mus1")>-1) {
                    String tmp[]=split(msgp,'!');
                    for(int i=0;i<3;i++)datamus[i]=PApplet.parseInt(tmp[i+1]);
                }
            }
        }
        if (!isloop) {
            break;
        }
        delay(1);
    }
}

boolean iscon = false; //соединение сейчас?

```



```

boolean isloop =false;//мы в параллельном потоке?
ArrayList parkovkab=new ArrayList();//кнопки
int mesta[]=new int[6];//состояние мест
long timerp =0;          //таймер обновление мест
int bpc1 = color(172, 0, 0); //цветовая схема
int bpc2 = color(200, 0, 0);
int bpc3 = color(230, 0, 0);
int bpc4 = color(172, 172, 0);
int bpc5 = color(200, 200, 0);
int bpc6 = color(230, 230, 0);
String msgp=""; //отладочно-информативная строка
Button bron, IT; //ещё кнопки
int cursorp=0; //курсор выбранного места
int lastcp = -1; //прошрое выбранное место
int mybron=-1; //забронированное место
String IDP=""; //ID для брони
public void initparkovka() {//инит
    Button b;
    b = new Button(dw/20, dw/20, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    b = new Button(dw/20, dw/10*3, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    b = new Button(dw/20, dw/20*11, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    b = new Button(dw/20*11, dw/20, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    b = new Button(dw/20*11, dw/10*3, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    b = new Button(dw/20*11, dw/20*11, dw/5*2, dw/5, bc1, bc2, bc3, "", bc4);
    parkovkab.add(b);
    bron=new Button(dw/4, dh/5*4, dw/2, dh/20, bc1, bc2, bc3, "Забронировать",
bc4);
    IT = new Button(dw/4, dh/10*9, dw/2, dh/20, bc1, bc2, bc3, "Я приехал!",
bc4);
    IDP=str(PApplet.parseInt(random(10000, 99999)));
    //mybron = data.getInt("mybron");
}
public void parkovka() {
    if (!isloop)//параллельный поток для обработки связи с сервером/парковкой
        thread("contrelload");
    updp();//графическое и взаимодействие обновление
}

public void contrelload() {//проверяет входящие данные и обрабатывает
    isloop=true;
    while (true) {
        if (c.available()>0) {
            String msgp=c.readStringUntil('\n');
            if (msgp!=null) {
                if (msgp.indexOf("!getpark!")==0) {
                    for (int i =0; i<mesta.length; i++) {
                        mesta[i]=PApplet.parseInt(msgp.substring(i+9, i+10));
                    }
                }
            }
        }
    }
    if (!isloop) {

```

```

        break;
    }
    delay(1);
}

}

public void contsend() { //бронировани
    c.write("spspsp"+str(mybron+1)+IDP+"\n");
    //data.setInt("mybron", mybron);
    //saveJSONObject(data, "./data/data.json");
}

public void updp() {
    Button b; //обновление кнопок
    for (int i =0; i<parkovkab.size(); i++) {
        b = (Button)parkovkab.get(i);
        b.up();
        if (mesta[i]==0) {
            b.c1=bc1;
            b.c2=bc2;
            b.c3=bc3;
            if (b.press()) {
                msgp="Это место свободно";
                if(lastcp == cursorp)lastcp=-1;
                else lastcp=cursorp;
                cursorp=i;
            }
        } else if (mesta[i]==1) {
            b.c1=bpc4;
            b.c2=bpc5;
            b.c3=bpc6;
            if (b.press()) {
                msgp="Это место забронировано";
                cursorp=i;
            }
        } else if (mesta[i]==2) {
            b.c1=bpc1;
            b.c2=bpc2;
            b.c3=bpc3;
            if (b.press()) {
                msgp="Это место занято";
                cursorp=i;
            }
        }
        b.up();
        if (cursorp==i && lastcp!=i) {
            fill(150, 200);
            noStroke();
            rect(b.x, b.y, b.w, b.h);
        }
    }
    bron.up();
    if (bron.press()) { //бронирование
        if (mesta[cursorp]==0) {
            mybron = cursorp;
            contsend();
        } else {

```

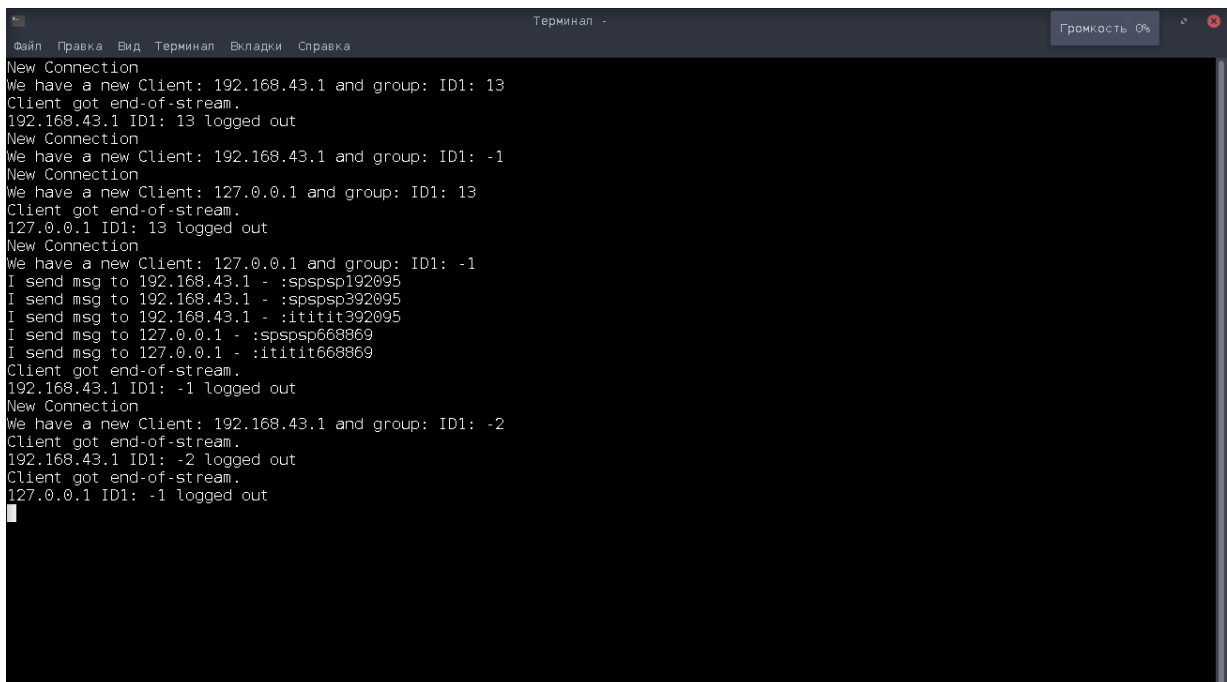
```

        msgp="Вы не можете забронировать это место";
    }
}
IT.up();
if (IT.press()) { //разбронирование
    c.write("ititit"+str(mybron+1)+IDP+"\n");
}
fill(255); //отладочно-информативная строка
textAlign(LEFT, TOP);
textSize(textsize);
text(msgp, 0, dw);
}

static public void main(String[] passedArgs) {
    String[] appletArgs = new String[] { "parkov" };
    if (passedArgs != null) {
        PApplet.main(concat(appletArgs, passedArgs));
    } else {
        PApplet.main(appletArgs);
    }
}
}
}

```

## Сервер:



```

Терминал
Файл Правка Вид Терминал Вкладки Справка
New Connection
We have a new Client: 192.168.43.1 and group: ID1: 13
Client got end-of-stream.
192.168.43.1 ID1: 13 logged out
New Connection
We have a new Client: 192.168.43.1 and group: ID1: -1
New Connection
We have a new Client: 127.0.0.1 and group: ID1: 13
Client got end-of-stream.
127.0.0.1 ID1: 13 logged out
New Connection
We have a new Client: 127.0.0.1 and group: ID1: -1
I send msg to 192.168.43.1 - :spspsp192095
I send msg to 192.168.43.1 - :spspsp392095
I send msg to 192.168.43.1 - :ititit392095
I send msg to 127.0.0.1 - :spspsp668869
I send msg to 127.0.0.1 - :ititit668869
Client got end-of-stream.
192.168.43.1 ID1: -1 logged out
New Connection
We have a new Client: 192.168.43.1 and group: ID1: -2
Client got end-of-stream.
192.168.43.1 ID1: -2 logged out
Client got end-of-stream.
127.0.0.1 ID1: -1 logged out

```

## Код

```
import processing.core.*;
import processing.data.*;
import processing.event.*;
import processing.opengl.*;

import java.lang.reflect.*;
import java.net.*;
import java.io.*;
import java.util.HashMap;
import java.util.ArrayList;
import java.io.File;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;

public class server_cons1 extends PApplet {
    static Server s=null; //сервер
    static ArrayList ClientList = new ArrayList(); //список клиентов
    static boolean Server=true; //сервер активен?
    static String msgs=""; //отладочная строка
    static boolean msgsb=false; //выводить отладочную строку сейчас?
    static public void main(String[] passedArgs) {
        server_loop();//основной цикл
    }
    public static void server_loop() {
        while (true) { //цикл
            if (Server && s==null) { //установка сервера
                s = new Server(48999);
            } else if (!Server && s!=null) {
                s.stop();
                s=null;
                ClientList.clear();
            }
            if (Server) { //если активны
                try {
                    listentoclients();//прослушиваем клиентов
                }
                catch(Exception ex) {
                }
            }
            if (msgsb) { //выводим отладочное сообщение
                println(msgs);
                msgsb=false;
            }
        }
    }
    public static void listentoclients() { //функция прослушивания клиентов
        Client NowClient = s.available(); //получаем "говорящего" клиента
        SmartClient newSmartClient = null; //создаем временный экземпляр класса
        "умного" клиента
        if (NowClient != null) { //если настоящий клиент есть
            for (int i =0; i<ClientList.size(); i++) { //
                SmartClient nnewSmartClient = (SmartClient)ClientList.get(i); //
                if (nnewSmartClient.thisclient == NowClient) { //
```

```

        newSmartClient = nnewSmartClient;
//Поиск клиента в списке зарегистрированных клиентов
        break;
    }
}
if (newSmartClient != null) { //если мы нашли
такого зарегистрированного клиента
    if (NowClient.available()>0) { //если есть
данные
        String msg=NowClient.readStringUntil('\n'); //считываем
данные
        if (msg.indexOf("?")>-1) { //если это
команда проверки соединения
            newSmartClient.thisclient.write("!!!!!!!\n");//то отправляем
ответ
            return; //выходим из
функции
        }
        for (int a = 0; a<ClientList.size(); a++) {
//иначе мы приняли сообщение
            SmartClient nnewSmartClient = (SmartClient)ClientList.get(a);
//и ищем клиентов одногруппников
            if (nnewSmartClient.group.equals(newSmartClient.group) &&
nnewSmartClient.thisclient != NowClient) { //и отправляем всем им
сообщение
                print("I send msg to " + nnewSmartClient.thisclient.ip()+"
- :"+msg);
                nnewSmartClient.thisclient.write(msg);
            }
        }
    }
}
} else makeclient(NowClient); //если этот клиент не зарегистрирован,
пытается зарегистрировать
}
return;//выходим
}

public static void makeclient(Client nc) { //регистрация
    if (nc.available()>0) { //если есть входящие данные
        StringList input = new StringList();//список входящих строк
        for (int a =0; a<2; a++) {
            input.append(nc.readStringUntil('\n'));//считываем строки
        }
        nc.clear(); //очищаем буфер
        if (input.get(0)==null || !input.get(0).equals("SYSSC: HELLO\
n"))return;//проверяем заголовок
        String tmpgroup = input.get(1); //получаем ID группы
        SmartClient nowNewSmartClient = new SmartClient(nc, tmpgroup);
//добавляем в список клиентов
        ClientList.add(nowNewSmartClient);
        print("We have a new Client: " + nc.ip() + " and group: " + tmpgroup);
        //nc.write("OK\n");
    }
}

```

```

        return;
    }

    public static void serverEvent(Server ns, Client nc) { //если подключается
клиент
        println("New Connection");
    }

    public static void disconnectEvent(Client myOldClient) { //если отключается
клиент
        for (int i = 0; i < ClientList.size(); i++) { //удаляем клиента из списка
            SmartClient tmpclient = (SmartClient)ClientList.get(i);
            if (tmpclient.thisclient == myOldClient) {
                println(tmpclient.thisclient.ip()+" "+tmpclient.group.substring(0,
tmpclient.group.length()-1)+" logged out");
                ClientList.remove(tmpclient);
            }
        }
    }

    static public class SmartClient { //класс клиента
        Client thisclient;
        String group;
        public SmartClient(Client newClient, String newGroup) {
            thisclient = newClient;
            group = newGroup;
        }
    }
}

```