Департамент образования мэрии города Новосибирска
Дворец творчества детей и учащейся молодежи «Юниор»

Открытый городской конкурс исследовательских проектов учащихся 5-8 классов

Направление: научно-техническое

Умная парковка

Авторы: **Корецкий Алексей Олегович** 8 класс,
Плющев Александр, 6 класс**,**
МБОУ Гимназия № 16 «Французская»,
Ленинский район, г. Новосибирск.

Руководитель проекта: Кузнецова Галина Вячеславовна, учитель физики

Контактный телефон руководителя: +7-923-777-05-849

г. Новосибирск, 2019

Паспорт проекта

Название проекта	«Умная парковка»
ФИО разработчика	Корецкий Алексей Олегович Плющев Александр
Класс	8в 6а
Название, № учебного учреждения	МБОУ Гимназия № 16 «Французская»
Предметная область	информатика, программирование, электроника
Время разработки проекта	ноябрь 2018 – март 2019г.
Проблема	Рост автомобилей в городе, что выливается в проблему их парковки
Цель	1. Изучить тему «парковки в городе» 2. Создать модель умной парковки.
Задачи	1. Сбор, исследование, анализ теории. 2. Выбор оптимального решения парковки.. 3. Разработка макета умной парковки. 4.Создание презентации «Разработка макета умной парковки»
Тип проекта	Поисковый, исследовательский, творческий.
Используемая технология	Мультимедиа, прототипирование (QCAD); программирование OpenScad (среда и язык), Arduino IDE(среда; язык С подобный), Processing IDE(среда; Java - язык)
Форма продукта проекта	Мультимедийная презентация и макет умной парковки.
Характеристика содержания проекта	В проекте изучены все известные типы парковок, проведен анализ их стоимости и эффективности., представлен процесс разработки макета умной парковки
Исследовательский компонент	Анализ все известных видов парковок
Область применения результата проекта:	Создание умных парковочных мест в городских условиях (и не только)
Результативность:	Презентация и модель умной парковки

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 «пятачках» стоят стеллажи из машин? А ажурные двадцатиэтажные 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 удается 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 м2

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. Суть решения состоит во внедрении в каждое парковочное место датчика, который определяет наличие или отсутствие транспортного средства. Далее микроконтроллер Ардуино, обрабатывает показания, изменяет состояние в своей памяти каждого места,устанавливает индикацию,светодиод горит — свободно, не горит — занято, мигает - забронировано, и передает информацию состояния на сервер. Сам контроллер Arduino будет иметь выход в глобальную сеть,с помощью wifi или ethernet модулей,через которую осуществляется 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) пироэлектрический 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 чуствительны. 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 доступа в глобальную сеть интернет будет использован модуль wifi esp8266. Вот итоговая принципиальная схема:

Программный — локальный

Программа для Arduino и телефона.
Arduino микроконтроллер просчитывает показания ультразвуковых датчиков по формуле:
L = delay/ 29 / 2

Где L — расстояние до объекта
delay — задержка ультразвукового импульса
/29 — скорость распространения звука в воздухе — см/мкс
/2 — звук проходит двойное расстояние — до объекта и обратно.
Использовано объектно ориентированное программирование — создан класс Mesto, объекты которого обрабатываю свой датчик, определяют состояние парковочного места и устанавливает индикацию.
Приложение для телефона реализует графический интерфейс для просмотра места и бронирования. Место серое — свободно, красное — занято, жёлтое — забронировано.
Использовалась библиотека Processing API которая предназначена для графики и как таковое GUI не поддерживает — писался класс кнопки для облегчения программирования.

Программный — серверный

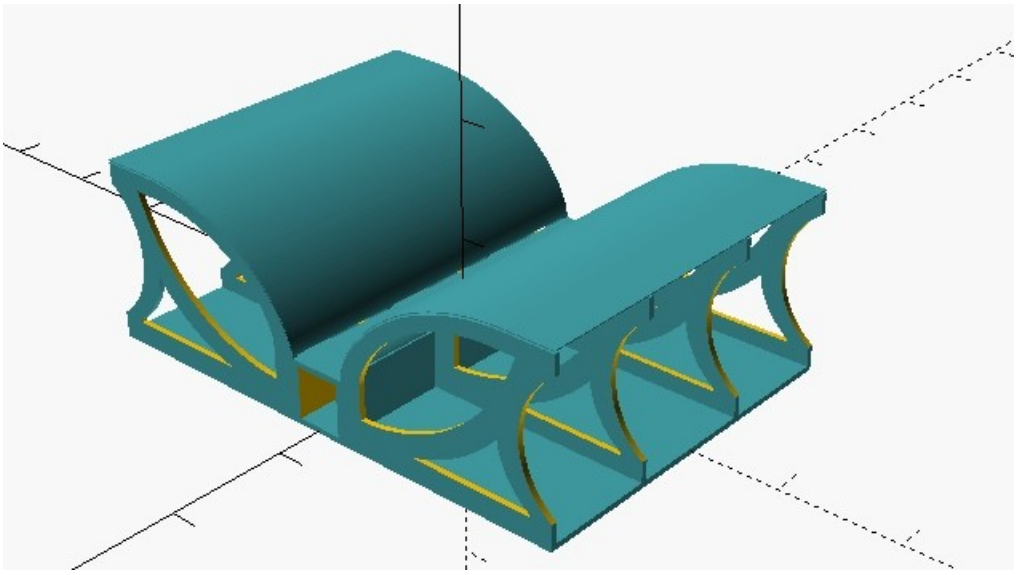
Сервер писался самостоятельно и больше для системы умного города, в которую входит и парковка, поэтому для облегчения совместимости выбран этот путь. Написана серверная программа в той же среде Processing IDE которая имеет удобные библиотеки реализации сервера и клиента.
Логика:

1) Клиент подключается
2) если присылает данные проверяем «зарегистрирован» ли он
3) если да то читаем данные проверяем на команды и отправляем всем соответствующей группе
   если нет читаем данные — удовлетворяют — «регистрируем», иначе удаляем.

«регистрация» - передача сведений о сервере и идентификаторе группе; пример:
SYSSC: HELLO\n
ID1: 13\n
В данном примере 13 — идентификатор группы. При отправки данных, они транслируются всем входящим в группу кроме отправителя.
К Arduino — прибавляется код соеденения с сервером, регистрацией и обработкой локальных парковочных команд.
К телефону — 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 кто то бронирует, он отправляет — setpark 0\n
где 0 — идентификатор места == индекс в масиве

Макет

Для реализации проекта был разработан дизайн парковки, предусматривающий защиту парковки от непогоды.
Моделирование производилось в программе OpenScad.



В дальнейшем детали были изготовлены лазерной резкой.

Код микроконтроллера:

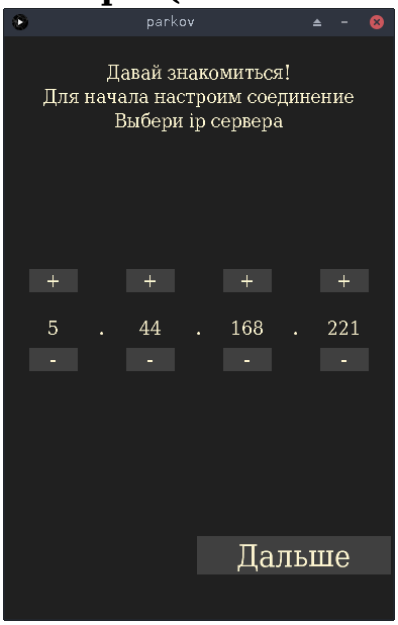
```
class Mesto {
  public:
    uint8_t ismesto = 0;
    uint8_t trig;
    uint8_t echo;
    uint8_t led;
    boolean state = false;
    unsigned long timer;
    int distance = 50;
    //    Mesto(uint8_t _trig, uint8_t _echo, uint8_t _led) {
    //      trig = _trig;
    //      echo = _echo;
    //      led = _led;
    //    }
    void begin(uint8_t _trig, uint8_t _echo, uint8_t _led) {
      trig = _trig;
      echo = _echo;
      led = _led;
      pinMode(trig, OUTPUT);
      pinMode(echo, INPUT);
      pinMode(led, OUTPUT);
    }
    void zabr() {
      ismesto = 1;
    }
    void up() {
      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 < 10)) {
        ismesto = 2;
      } else if (ismesto != 1) {
        ismesto = 0;
      }
      switch (ismesto) {
        case 2:
          state = false;
          break;
        case 1:
          if (millis() - timer > 300) {
            state = !state;
            timer = millis();
          }
          break;
        case 0:
          state = true;
```

```
        break;
      default:
        break;
    }
    digitalWrite(led, state);
  }
};

Mesto ms[6];
uint8_t pins[6][3] = {
  {6, 7, 13},
  {1, 1, 1},
  {1, 1, 1},
  {1, 1, 1},
  {1, 1, 1},
  {1, 1, 1}
};

#include <SoftwareSerial.h>
SoftwareSerial esp(9, 8);

unsigned long t = 0;
unsigned long t1 = 0;
char parkstate[] = "000100";
uint8_t mesta[6];
void setup() {
  for (int i = 0; i < 6; i++) {
    ms[i].begin(pins[i][0], pins[i][1], pins[i][2]);
  }
  esp.begin(115200);
  while (!esp);
  reconnect();
  ms[0].ismesto = 1;
  Serial.begin(9600);
  esp.flush();
  t = millis();
}

void loop() {
  // put your main code here, to run repeatedly:
  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 - clear, 1 - zabr, 2 - zani

  if (esp.available() > 0) {
    String msg = esp.readStringUntil('\n');
```

```
    if (msg.indexOf("setpark") > -1) {
      int i = msg.substring(7, msg.length() - 1).toInt();
      ms[i].ismesto = 1;
    }
  }

  if (millis() - t > 60000) {
    esp.flush();
    esp.print("?\n");
    String tmp = "";
    while (true) {
      if (millis() - t > 5000) {
        reconnect();
      }
      if (esp.available() > 0) {
        tmp += (char)esp.read();
      }
      if (tmp.indexOf("!") > -1) {
        break;
      }
    }
    t = millis();
  }
}
void reconnect() {
  esp.print("+++");
  delay(500);
  if (!sendcmd("AT", "OK", 250))return;
  if (!sendcmd("AT+CWMODE=1", "OK", 250))return;
  if (!sendcmd("AT+CWJAP=\"ROSTELEKOM\",\"lenovozuk123\"", "OK",
7000))return;
  if (!sendcmd("AT+CIPSTART=\"TCP\",\"192.168.1.3\",1881", "OK",
5000))return; //94.180.117.139
  if (!sendcmd("AT+CIPMODE=1", "OK", 250))return;
  esp.print("AT+CIPSEND\r\n");
  delay(200);
  esp.print("SYSSC: HELLO\nID1: -1\n");
  Serial.println("Donw?!");
  t = millis();
}
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 processing.core.*;
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;
int dw, dh;
ArrayList hellob=new ArrayList();
int bc1 = color(64);
int bc2 = color(96);
```

```
int bc4 = color(0xFE, 0xF5, 0xD0);
int bc3 = color(128);
int state=0;
PFont myfont;
int textsize;
public void settings() {
  float k=0.8f;
  size(PApplet.parseInt(480*k), PApplet.parseInt(720*k));
}

public void setup() {
  orientation(PORTRAIT);
  background(32);
  dw=width;
  dh=height;
  myfont=loadFont("DejaVuSerifCondensed-64.vlw");
  textFont(myfont);
  textsize=dh/30;
  fill(255);
  textSize(textsize);
  inithello();
  initcard();
  initparkovka();
}

public void draw() {
  background(32);
  switch(state) {
  case -2:
    delay(6000);
    exit();
    break;
  case -1:
    textAlign(CENTER);
    text("\nИзвените,\nчто то пошло не так...\n(1)проверьте интернет
соединение\n(2)проверьте правельность ip\n(3)попробуйте позже\nдо
встречи...", dw/2, textsize);
    state = -2;
    break;
  case 0:
    hello();
    break;
  case 1:
    break;
    case 2:
    card();
    break;
    case 3:
    parkovka();
    break;
  }
}
```

```
public void keyPressed() {
  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(){
  state = 2;
}
class Button {
  int x, y, w, h;
  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;
String obm[] = {"Мусорка", "Сервер", "Парковка", "Дом"};
ArrayList cardb=new ArrayList();
int cursorv=0;
public void initcard() {
  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 = "";
public void card() {
```

```
  image(maps, 0, 0);
  textSize(textsize);
  textAlign(LEFT, TOP);
  text("Выбор: "+obm[cursorv]+"\n"+msgc, 0, dw+textsize);
  for (int i =0; i<cardb.size(); i++) {
    Button b = (Button)cardb.get(i);
    b.up();
    if (b.press()) {
      if (i==2) {
        msgc="";
        state = 3;
      } else if (i==1) {
        msgc="Вы не имеете прав";
      } else {
        msgc="Данный раздел пока не поддерживается\nПросим
извинения.\nПриложение в разработке";
      }
    }
  }
}
//hello
int cursor=0;
int ip[]={5, 44, 168, 221};

public void inithello() {
  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;
public void hello() {
  if (!keyb) {
    //openKeyboard();
    keyb=true;
  }
```

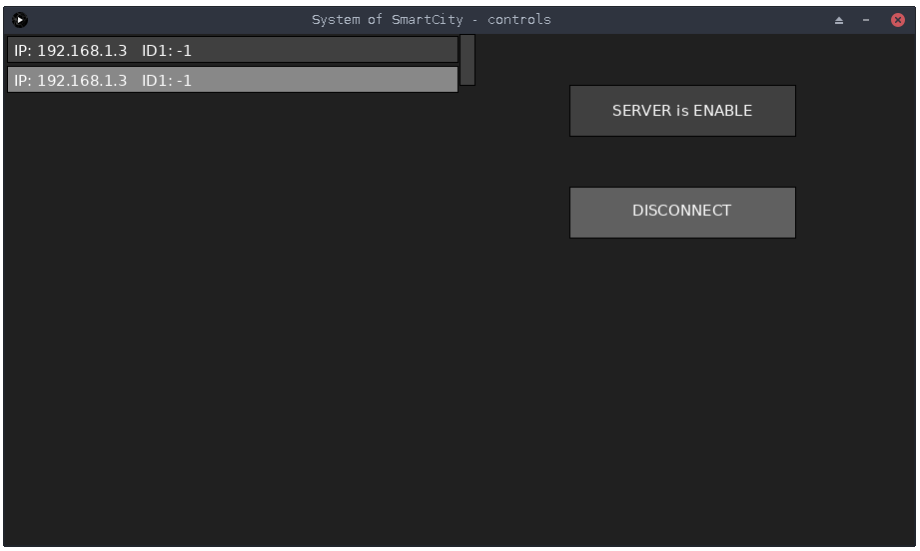
```
  textAlign(CENTER);
  text("\nДавай знакомиться!\nДля начала настроим соединение\
nВыбери ip сервера", dw/2, textsize);
  for (int i =0; i<hellob.size(); i++) {
    Button b = (Button)hellob.get(i);
    b.up();
    if (i==8) {
      if (b.press()) {
        if (keyb) {
          //closeKeyboard();
          keyb=false;
        }
        connect("13", 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);
  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) {
  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]),
1881);
      if (c==null || !c.active() || c.ip() == null) {
        state = -1;
        return;
      }
      c.write("SYSSC: HELLO\nID1: "+group+"\n");
    }
  }
  catch(Exception ex) {
    state = -1;
    return;
```

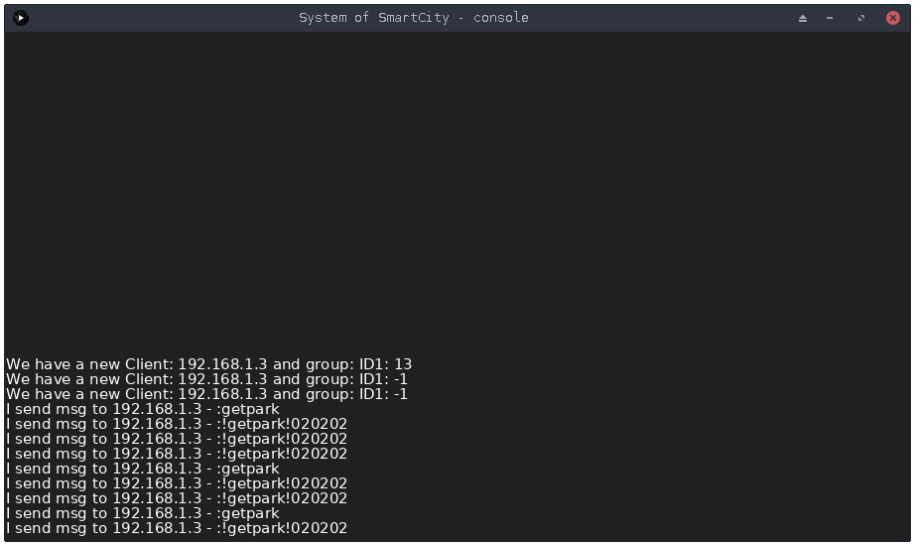
```
  }
  state=nstate;
}
boolean iscon = false;
boolean isloop =false;
ArrayList parkovkab=new ArrayList();
int mesta[]=new int[6];
long timerp =0;
int bpc1 = color(172, 0, 0);
int bpc2 = color(200, 0, 0);
int bpc3 = color(230, 0, 0);
int bpc4 = color(172, 172, 0);
int bpc5 = color(200, 200, 0);
int bpc6 = color(230, 230, 0);
String msgp="";
Button bron;
int cursorp=0;
int mybron=-1;
public void initparkovka() {
  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);
}
public void parkovka() {
  if (!isloop)
    thread("contreload");
  updp();
}

public void contreload() {
  isloop=true;
  while (true) {
    if (!iscon) {
      c.stop();
      connect("-1", 3);
      delay(100);
      iscon=true;
    }
    if (c.available()>0) {
      String msgp=c.readStringUntil('\n');
```

```
      if (msgp!=null) {
        print(msgp);
        if (msgp.indexOf("!getpark!")==0) {
          for (int i =0; i<mesta.length; i++) {
            mesta[i]=PApplet.parseInt(msgp.substring(i+9, i+10));
            print(mesta[i]);
          }
          println();
        }
      }
    }
    if (!isloop) {
      break;
    }
    delay(1);
  }
}

public void contsend() {
  if (mybron>=0)
    c.write("setpark"+mybron+"\n");
}

public void updp() {
  Button b;
  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";
        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;
      }
```

```
    }
  }
  bron.up();
  if (bron.press()) {
    if (mesta[cursorp]==0) {
      mesta[cursorp]=1;
      mybron = cursorp;
      contsend();
    } else {
      msgp="Вы не можете забронировать это место";
    }
  }
  fill(255);
  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:





Код

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

import processing.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 server_for_house extends PApplet {


Server s=null;
ArrayList ClientList = new ArrayList();
```

```
boolean Server=true;
console cns;
boolean con=true;
JSONObject setting;
int textsize;
int move=0, tomove=0;
int xselect=-50, yselect=-50;
SmartClient clientselect=null;
boolean movetomove=false;
public void settings() {
  size(displayWidth*2/3, displayHeight*2/3);
}
public void setup() {
  surface.setTitle("System of SmartCity - controls");
  setting=loadJSONObject("./data/setting.json");
  con = setting.getBoolean("console");
  if (con) {
    cns = new console();
  } else cns=null;
  textsize=displayHeight/50;
  textSize(textsize);
  thread("server_loop");
}
public void draw() {
  background(32);
  drawlist();
  drawcontrolmenu();
  delay(1);
}
public void printToScr(String text) {
  //print(text);
  if (con) {
    cns.texttoscr=text;
    cns.texT=true;
  }
  return;
}


boolean push=true;
public void drawcontrolmenu() {
  stroke(0);
  boolean now=mousePressed;
  if (mouseX>width/8*5 && mouseY>height/10 && mouseX<width/8*7 &&
mouseY<height/10*2 && now && !push) {
    fill(128);
    Server=!Server;
  } else if (mouseX>width/8*5 && mouseY>height/10 &&
mouseX<width/8*7 && mouseY<height/10*2 ) {
    fill(96);
  } else {
    fill(64);
```

```
  }
  push=now;
  rect(width/8*5, height/10, width/8*2, height/10);
  String state;
  if (Server)state="ENABLE";
  else state="DISABLE";
  textAlign(CENTER, CENTER);
  fill(255);
  text("SERVER is "+state, width/8*6, height/20*3);

  if (mouseX>width/8*5 && mouseY>height/10*3 && mouseX<width/8*7
&& mouseY<height/10*4 && mousePressed) {
    fill(128);
    if (clientselect!=null && ClientList.indexOf(clientselect)>-1) {
      ClientList.remove(clientselect);
    }
  } else if (mouseX>width/8*5 && mouseY>height/10*3 && mouseX<width/
8*7 && mouseY<height/10*4 ) {
    fill(96);
  } else {
    fill(64);
  }
  rect(width/8*5, height/10*3, width/8*2, height/10);
  textAlign(CENTER, CENTER);
  fill(255);
  text("DISCONNECT", width/8*6, height/20*7);

  textAlign(LEFT, BASELINE);
}

public void server_loop() {
  while (true) {
    if (Server && s==null)s = new Server(this, 1881);
    else if (!Server && s!=null) {
      s.stop();
      s=null;
    }
    if (Server) {
      try{
      listentoclients();
      }catch(Exception ex){}
    }
    delay(1);
  }
}

public void drawlist() {
  for (int i=0; i<ClientList.size(); i++) {
    SmartClient nowclient=(SmartClient)ClientList.get(i);
    if (nowclient.thisclient.ip()==null) {
      ClientList.remove(nowclient);
      continue;
```

```
    }
    if ((i+1)*textsize*2+move<0)continue;
    if (mouseX>2 && mouseX<width/2-2 && mouseY>textsize*2*i+2+move
&& mouseY<textsize*2*(i+1)-2+move && mousePressed && !movetomove)
{
      clientselect=(SmartClient)ClientList.get(i);
      xselect=2;
      yselect=textsize*2*i+2;
    }
    if (mouseX>2 && mouseX<width/2-2 && mouseY>textsize*2*i+2+move
&& mouseY<textsize*2*(i+1)-2+move)
      fill(96);
    else fill(64);
    stroke(0);
    rect(2, textsize*2*i+2+move, width/2-4, textsize*2-4);
    fill(255);
    text(" IP: "+nowclient.thisclient.ip()+"   "+nowclient.group, 4,
textsize*3/2+textsize*2*i+move);
    if (i*textsize*2+move>height)break;
  }
  fill(255, 96);
  stroke(0);
  if (ClientList.indexOf(clientselect) > -1 ) {
    rect(xselect, yselect+move, width/2-4, textsize*2-4);
  }
  if (mouseX>width/2 && mouseY>tomove && mouseX<width/60*31 &&
mouseY<height/15+tomove) {
    fill(96);
  } else {
    fill(64);
  }
  if (mouseX>width/2 && mouseY>tomove && mouseX<width/60*31 &&
mouseY<height/10+tomove && mousePressed) {
    movetomove=true;
  }
  if (movetomove) {
    tomove=mouseY-height/30;
    if (tomove<0)tomove=0;
    if (tomove>height/10*9)tomove=height/10*9;
    move=(int)map(tomove, 0, height/10*9, 0, -(ClientList.size()-
1)*textsize*2+height);
    if (move>0) {
      move=0;
    } else if (ClientList.size()*textsize*2>height && -(ClientList.size()-
1)*textsize*2+height>move) {
      move=-(ClientList.size()-1)*textsize*2+height;
    } else if (ClientList.size()*textsize*2<height) {
      move=0;
    }
  }
  if (!mousePressed)movetomove=false;
  rect(width/2, tomove, width/60, height/10);
```

```
}

public void listentoclients() {
  Client NowClient = s.available();
  SmartClient newSmartClient = null;
  if (NowClient != null) {
    for (int i =0; i<ClientList.size(); i++) {
      SmartClient nnewSmartClient = (SmartClient)ClientList.get(i);
      if (nnewSmartClient.thisclient == NowClient) {
        newSmartClient = nnewSmartClient;
        break;
      }
    }
    if (newSmartClient != null) {
      if (NowClient.available()>0) {
        String msg=NowClient.readStringUntil('\n');
        if(msg.indexOf("?")>-1){
          newSmartClient.thisclient.write("!\n");
          return;
        }
        for (int a = 0; a<ClientList.size(); a++) {
          SmartClient nnewSmartClient = (SmartClient)ClientList.get(a);
          if (nnewSmartClient.group.equals(newSmartClient.group) &&
nnewSmartClient.thisclient != NowClient) {
            printToScr("I send msg to " + nnewSmartClient.thisclient.ip()+"
- :"+msg);
            nnewSmartClient.thisclient.write(msg);
          }
        }
      }
    } else makeclient(NowClient);
  }
  return;
}

public void makeclient(Client nc) {
  long timer1 =millis();
  while (millis()-timer1<100) {
    if (nc.available()>0) {
      StringList input = new StringList();
      for (int a =0; a<2; a++) {
        input.append(nc.readStringUntil('\n'));
      }
      nc.clear();
      if (input.get(0)==null || !input.get(0).equals("SYSSC: HELLO\n"))return;
      String tmpgroup = input.get(1);
      SmartClient nowNewSmartClient = new SmartClient(nc, tmpgroup);
      ClientList.add(nowNewSmartClient);
      printToScr("We have a new Client: " + nc.ip() + " and group: " +
tmpgroup);
      //nc.write("OK\n");
      return;
```

```
    }
  }
  return;
}

public void serverEvent(Server ns, Client nc) {
  printToScr("New Connection");
}

public void disconnectEvent(Client myOldClient) {
  for (int i =0; i< ClientList.size(); i++) {
    SmartClient tmpclient = (SmartClient)ClientList.get(i);
    if (tmpclient.thisclient == myOldClient) {
      printToScr(tmpclient.thisclient.ip()+" "+tmpclient.group.substring(0,
tmpclient.group.length()-1)+" logged out");
      ClientList.remove(tmpclient);
    }
  }
}

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

public void keyPressed() {
  if (key == ESC) {
    s.stop();
    exit();
  }
}
public void mouseWheel(MouseEvent event) {
  float e = event.getCount();
  move-=e*5;
  if (move>0) {
    move=0;
  } else if (ClientList.size()*textsize*2>height && -(ClientList.size()-
1)*textsize*2+height>move) {
    move=-(ClientList.size()-1)*textsize*2+height;
  } else if (ClientList.size()*textsize*2<height) {
    move=0;
  }
  tomove=(int)map(move, 0, -(ClientList.size()-1)*textsize*2+height, 0,
height/10*9);
}
class console extends PApplet {
  PImage scr;
  boolean texT=false;
```

```
  String texttoscr="";
  int textsize;
  public console() {
    super();
    PApplet.runSketch(new String[]{this.getClass().getName()}, this);
  }
  public void settings() {
    size(displayWidth*2/3, displayHeight*2/3);
  }
  public void setup() {
    surface.setTitle("System of SmartCity - console");
    background(32);
    surface.setResizable(true);
    textsize=displayHeight/50;
    textSize(textsize);
    scr=get();
  }
  public void draw() {
    if (texT) {
      background(32);
      image(scr, 0, height-scr.height-textsize);
      text(texttoscr, 0, height-textsize/2);
      scr=get();
      texT=false;
    }
    image(scr,0,height-scr.height);
  }
}
  static public void main(String[] passedArgs) {
    String[] appletArgs = new String[] { "server_for_house" };
    if (passedArgs != null) {
      PApplet.main(concat(appletArgs, passedArgs));
    } else {
      PApplet.main(appletArgs);
    }
  }
}
```