**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**Факультет** Инженерный

**Кафедра** Информационных систем и технологий

**Дата регистрации работы в деканате \_\_\_\_\_\_\_\_\_**

**Дата регистрации работы на кафедре \_\_\_\_\_\_\_\_\_**

**Отметка о допуске к защите \_\_\_\_\_\_\_\_\_**

**Оценка за защиту \_\_\_\_\_\_\_\_\_**

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине:** «Программирование сетевых приложений»

**Тема:**

**«Умный дом»**

**Исполнитель:**

**студент (факультет, курс, группа)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**фамилия, имя, отчество**

**Руководитель:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**наученое звание, ученая степень, должность**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**фамилия, имя, отчество**

**Барановичи 20****22**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc121607522)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc121607523)

[1.1. Постановка задачи 5](#_Toc121607524)

[1.2. Описание архитектуры приложения 5](#_Toc121607525)

[1.3. Описание использованных в разработке компонентов 6](#_Toc121607526)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 12](#_Toc121607527)

[2.1. Руководство пользователя 12](#_Toc121607528)

[2.2. Тестирование 13](#_Toc121607529)

[ЗАКЛЮЧЕНИЕ 20](#_Toc121607530)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 21](#_Toc121607531)

[ПРИЛОЖЕНИЯ 22](#_Toc121607532)

# ВВЕДЕНИЕ

За последнее десятилетие приобрели популярность интеллектуальные системы умного дома. Умный дом — это высокотехнологичная система, позволяющая объединить все коммуникации в одну и поставить её под управление программы, программируемой и настраиваемой под все потребности, и пожелания хозяина. Подобные системы добавляют комфорт в жизнь и сводят потребность обслуживать коммуникации и приборы к минимуму.

На текущий момент производством и продажей умных домов занимается большое количество компаний, начиная с крупных транснациональных корпораций, заканчивая частными организациями. Главное отличие между ними состоит в том, что решения частных компаний являются проводными, поэтому их нужно устанавливать либо сразу во время ремонта, либо делать его под саму систему. Крупные же компании чаще всего производят линейку разных «умных» приборов, которое могут объединятся и взаимодействовать друг с другом.

Однако, у обоих вариантов имеется существенный недостаток – их стоимость. Решения частных компаний – из-за вышеназванных особенностей, решений крупных – из-за необходимости приобретать несколько приборов сразу, чтобы создать систему, способную обеспечить требования и пожелания пользователя, которые сами, в свою очередь, стоят выше, нежели их «неумные» аналоги. В связи с этим, хотя подобные системы и являются крайне полезными для людей, они являются недоступными для большинства.

В то же время, подобные системы зачастую перегружены ненужными для конкретного пользователя функциями, результате чего человек платит высокую цену за ненужные ему функции.

Однако, подобную систему возможно создать своими руками, что исключит как ненужный функционал, так и высокую стоимость.

Целью данного курсового проекта является развитие навыков реализации сетевых приложений с использованием прикладного интерфейса программирования сокетов; средой разработки прикладных сетевых приложений; получение опыта разработки сетевых программных средств.

Задачей курсового проекта является создание наглядного примера значительно более дешёвой системы умного дома.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

* 1. **Постановка задачи**

Для выполнения поставленной задачи необходимо:

* Создать приложение, способное отправлять команды управляющему контроллеру через локальную Wi-Fi сеть;
* Написать программу для контроллера, принимающую команды через локальную Wi-Fi сеть;
* Создать наглядный макет, позволяющую увидеть выполнение команд.

Данная система позволит на расстоянии управлять электронными компонентами, что является хорошей основой для дальнейших разработок и исследований в данной направлении.

Для решения поставленной задачи потребуются навыки разработки мобильных сетевых приложений и навыки работы с микроконтроллерами их электронными компонентами.

* 1. **Описание архитектуры приложения**

Данное приложение было построено, используя шаблон «Наблюдатель», так же известный как «Подчинённые». Данный шаблон позволяет реализовать механизм, который позволяет одному объекту получать оповещения об изменении состояния других объектов, тем самым как бы «наблюдать» за ними. Шаблон Наблюдатель определяет зависимость «один-ко-многим» между объектами, так что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются.

Шаблон «наблюдатель» применяется в тех случаях, когда система обладает следующими свойствами:

* существует как минимум один объект, рассылающий сообщения;
* имеется не менее одного получателя сообщений, причём их количество и состав могут изменяться во время работы приложения;

Данный шаблон позволяет избежать [сильного зацепления](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D1%86%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) взаимодействующих классов и используется в основном в тех ситуациях, когда отправителя сообщений не интересует, что получатели будут делать с предоставленной им информацией.

Классы, на события которых другие классы подписываются, называются субъектами, а подписывающиеся классы называются наблюдателями.

В качестве субъекта в данном проекте выступает Android приложение, а в качестве наблюдателя – микроконтроллер. Так как микроконтроллер всего один, то в разработанном проекте данный шаблон реализует зависимость не «один-ко-многим», как было выше упомянуто, а «один к одному».

**1.3. Описание использованных в разработке компонентов**

Для разработки приложения была выбрана мобильная платформа с операционной системой Android в связи с огромным её распространением среди пользователей на сегодняшний день.

Для написания мобильного приложения была выбрана среда Android Studio— интегрированная среда разработки (IDE) для работы с платформой Android. Это программа для разработки мобильных приложений, поддерживающая несколько языков программирования, включая С/С++ и Java, имеет встроенный эмулятор с большим количеством настроек и большую библиотеку со всевозможными шаблонами и компонентами, существенно упрощающими и ускоряющими процесс разработки приложений для мобильных устройств.

Также для написания приложения была использована библиотека «OkHttp» – эффективный HTTP-клиент для приложений Android и Java. Он является альтернативой встроенным в SDK API, менее функциональным и сложным в применении. Так как разработанная система будет работать только в локальной сети, не было необходимости использовать HTTPS протокол, что делает данный выбор наиболее полезным в данной ситуации.

В качестве управляющего контроллера была выбрана платформа NodeMCU на основе микроконтроллера ESP8266. Работа с ним полностью посвящена работе с сетью Wi-Fi. Модули ESP8266 могут работать как станция (STA), благодаря чему мы можем подключить их к сети Wi-Fi. Они также могут работать как программная точка доступа (soft-AP) для создания собственной сети Wi-Fi. Когда модуль ESP8266 работает как программная точка доступа, мы можем подключать другие станции к модулю ESP. ESP8266 также может работать как в режиме станции, так и в режиме программной точки доступа одновременно. Это обеспечивает возможность построения, например, ячеистых сетей.

Для написания управляющей программы для микроконтроллера была использована Arduino IDE — интегрированная среда разработки для Windows, MacOS и Linux, разработанная на Си и C++, предназначенная для создания и загрузки программ на Arduino-совместимые платы, а также на платы других производителей.

Для исправной работы задуманной системы к программе микроконтроллера необходимо подключить библиотеку «ESP8266WiFi». Данная библиотека предоставляет ряд классов, позволяющих подключить сеть к микроконтроллеру и в дальнейшем работать с данной сетью.

Макет состоит из трёх светодиодов и шагового двигателя, который показывает управление рольшторой либо жалюзи. Внешний вид макета представлен на рисунке 1.1.

Схема подключения данного макета показана на рисунке 1.2.

Место для фотографии макета)

Дайте двигатель)

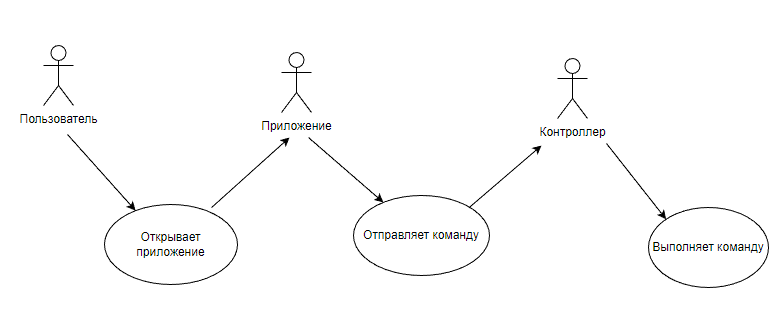
**Рисунок 1.1 – Внешний вид макета**

Место для схемы)

Дайте двигатель)

**Рисунок 1.2 – Схема подключения макета**

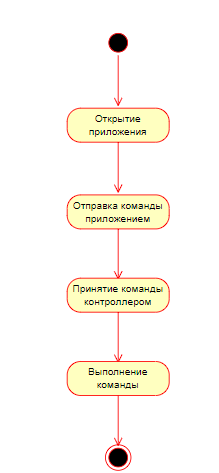
Диаграмма Use Case (Вариант использования) описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату. Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать (именно что, а не как).

Диаграмма Use Case представлена на рисунке 1.3.

**Рисунок 1.3 – Диаграмма Use Case**

Диаграмма деятельности (activity diagram) – [UML](https://ru.wikipedia.org/wiki/UML)-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых.

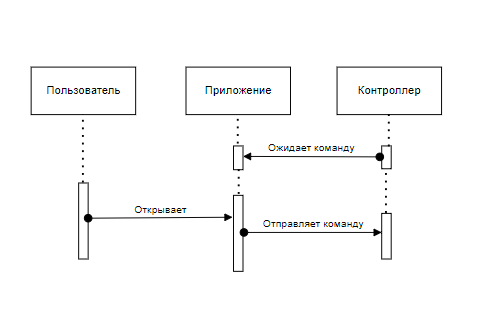
Диаграмма Activity представлена на рисунке 1.4.



**Рисунок 1.4 – Диаграмма Activity**

Диаграмма последовательности ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) sequence diagram) – [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

Диаграмма Sequence представлена на рисунке 1.5.



**Рисунок 1.5 – Диаграмма Sequence**

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

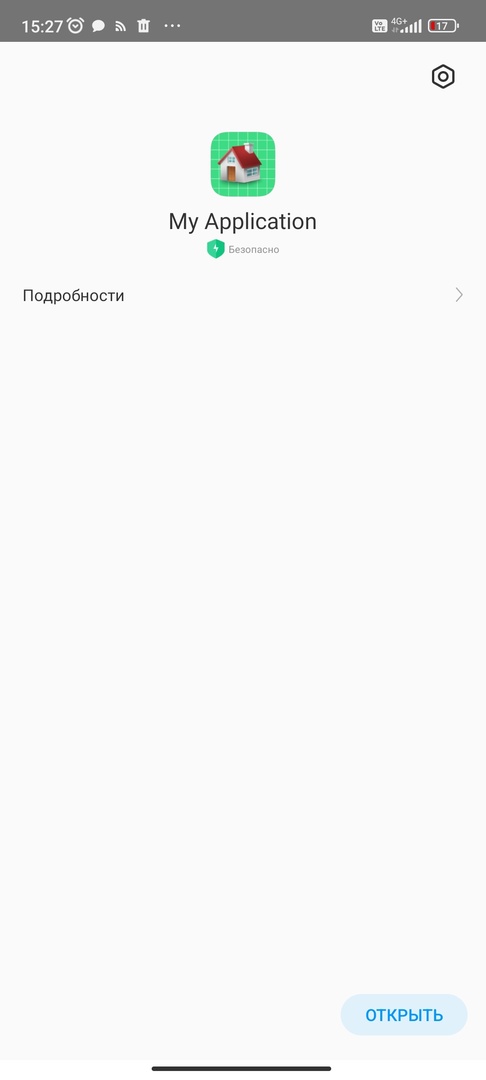
**2.1. Руководство пользователя**

Данное приложение поддерживает версии системы от Android 6.0 и выше.

Для того, чтобы установить программу, необходимо перейти по ссылке «https://disk.yandex.ru/d/07Dxfxs-0Mt0mg» и скачать файл MuHouse.apk (рисунок 2.1)

**Рисунок 2.1 – Скачивание файла программы**

После этого необходимо установить приложение на мобильное устройство (рисунок 2.2).



**Рисунок 2.2 – Установка программы на устройство**

**2.2. Тестирование**

После открытия приложения пользователь видит окно, изображённое на рисунке 2.3.



**Рисунок 2.3 – Внешний вид окна программы**

Для управления микроконтроллером необходимо подключить и его, и мобильное устройство к одной сети Wi-Fi. После этого в поле, изображённое на рисунке 2.4 необходимо ввести IP-адрес микроконтроллера и нажать на кнопку, показанную на рисунке 2.5), чтобы не вводить адрес каждый раз при использовании приложения (при подключении к одной сети IP-адрес контроллера будет оставаться неизменным, и введённый в поле IP-адрес будет действительным).



**Рисунок 2.4 – Поле для ввода IP-адреса**



**Рисунок 2.5 – Кнопка для сохранения IP-адреса**

После этого соединение между устройством и контроллером будет установлено.

Для отправки команд для отправки включения/выключения светодиодов необходимо нажать на одну из трёх кнопок, изображённых на рисунке 2.6. Первое нажатие на каждую отдельную кнопку включает соответствующий ей светодиод, второе – выключает.



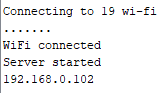
**Рисунок 2.6 – Кнопки для включения/выключения светодиодов**

За управление шаговым двигателем отвечают кнопки, изображённые на рисунке 2.7. Кнопка «Вверх» отвечает за его вращение в одну сторону, кнопка «Вниз», соответственно, за вращение в противоположную.

****

**Рисунок 2.7 – Кнопки для управления шаговым двигателем.**

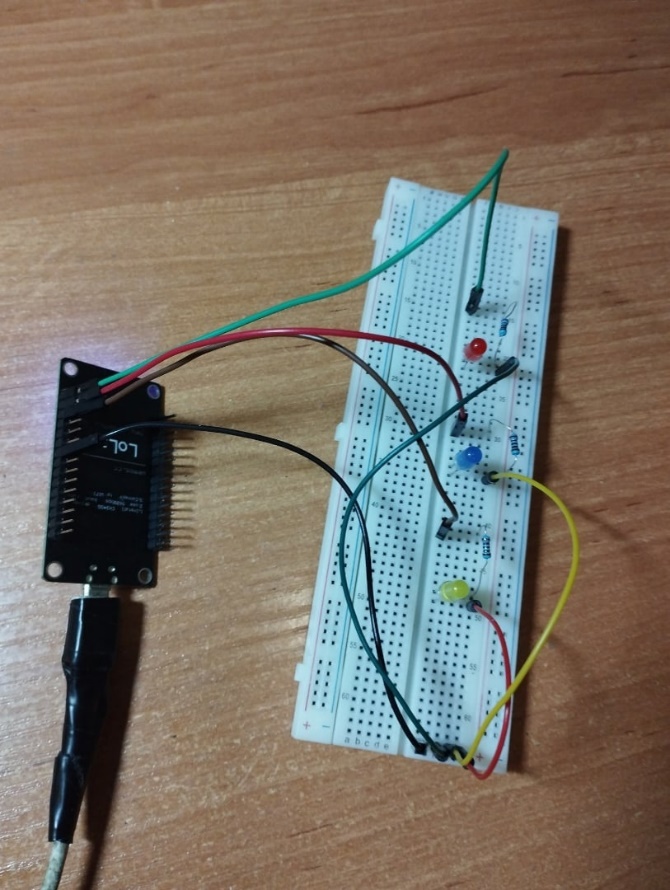
Последовательный пример использования приложения для работы со светодиодами (двигатель использоваться не будет, т.к. его вращение невозможно запечатлеть на рисунках) показан на рисунках 2.8 – 2.13.



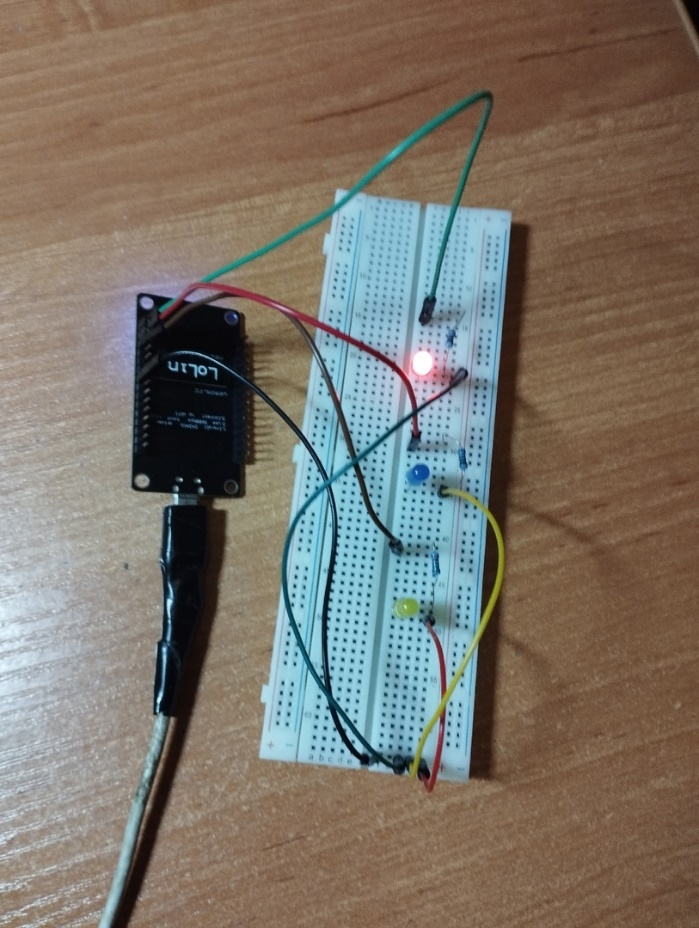
**Рисунок 2.8 – Определение IP-адреса в Arduino IDE**



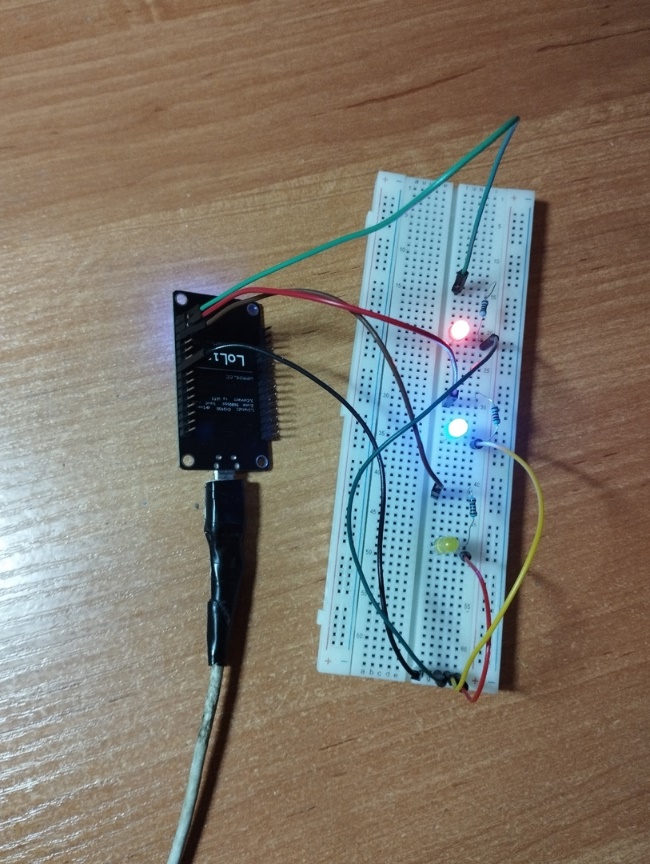
**Рисунок 2.9 – Ввод полученного IP-адреса в соответствующее поле**



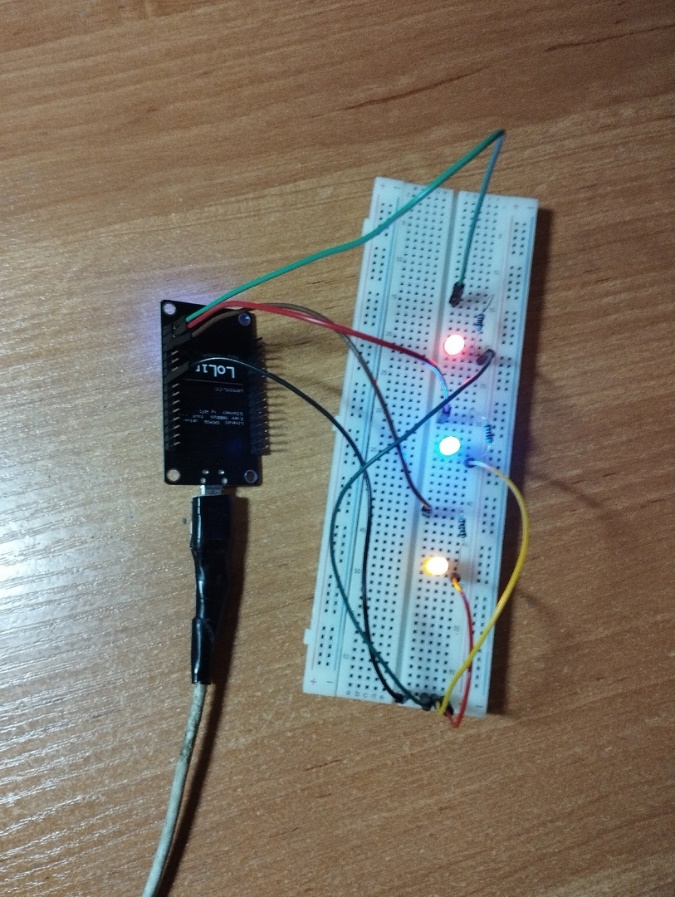
**Рисунок 2.10 – Первоначальное состояние макета**



**Рисунок 2.11 – Состояние макета после нажатия кнопки «Led1»**



**Рисунок 2.12 – Состояние макета после последующего нажатия кнопки «Led2»**



**Рисунок 2.13 – Состояние макета после последующего нажатия кнопки «Led3»**

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы было разработано мобильное сетевое программное средство для отправки команд на контроллер и управления электронными компонентами через сеть Wi-Fi. Разработанное приложение реализовано с использованием языка программирования Java в среде Android Studio.

Таким образом, поставленные цели были достигнуты, задачи работы были выполнены: создать наглядный пример более дешёвой относительно готовых решений системы «умного дома».

Проведенное тестирование программы позволяет сделать вывод о работоспособности программы и ее соответствии заданию. Выявленные недостатки предполагают дальнейшую работу над совершенствованием программы.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ESP8266. [Электронный ресурс]. –Режим доступа: https://ru.wikipedia.org/wiki/ESP8266. Дата доступа: 12.11.2020
2. Библиотека ESP8266WiFi. [Электронный ресурс]. –Режим доступа: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html. Дата доступа: 15.11.2020
3. Meet Android Studio [Электронный ресурс]. –Режим доступа: https://developer.android.com/studio/intro. Дата доступа: 25.10.2020
4. Наблюдатель (шаблон проектирования) [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C\_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD\_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Дата доступа: 02.11.2020.
5. Какую библиотеку работы с HTTP в Android выбрать? [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/281965/.Дата доступа: 25.10.2020

# ПРИЛОЖЕНИЯ

**ПРИЛОЖЕНИЕ А**

#include <ESP8266WiFi.h>

#include<Stepper.h>

bool led1IsOn = false;

bool led2IsOn = false;

bool led3IsOn = false;

const int LED\_0 = 16;

const int LED\_1 = 5;

const int LED\_2 = 4;

const int stepsPerRevolution = 200;

const char\* ssid = "19 wi-fi";

const char\* password = "6361735a";

// Create an instance of the server

// specify the port to listen on as an argument

WiFiServer server(80);

Stepper myStepper(stepsPerRevolution, 0, 2, 14, 12);

void setup() {

Serial.begin(9600);

pinMode(LED\_0, OUTPUT);

pinMode(LED\_1, OUTPUT);

pinMode(LED\_2, OUTPUT);

delay(1000);

digitalWrite(LED\_0, HIGH);

digitalWrite(LED\_1, HIGH);

digitalWrite(LED\_2, HIGH);

delay(1000);

digitalWrite(LED\_0, LOW);

digitalWrite(LED\_1, LOW);

digitalWrite(LED\_2, LOW);

delay(10);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

// Start the server

server.begin();

Serial.println("Server started");

// Print the IP address

Serial.println(WiFi.localIP());

}

void loop() {

// Check if a client has connected

WiFiClient client = server.available();

if (!client) {

return;

}

// Wait until the client sends some data

Serial.println("new client");

while(!client.available()){

delay(1);

}

// Read the first line of the request

String req = client.readStringUntil('\r');

Serial.println(req);

// Match the request

controller(req, client);

client.flush();

// Prepare the response

String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";

// Send the response to the client

client.print(s);

delay(1);

Serial.println("Client disonnected");

// The client will actually be disconnected

// when the function returns and 'client' object is detroyed

}

void controller(String req, WiFiClient client){

if(req.indexOf("/led1") != -1){

setLedState(LED\_0, led1IsOn);

led1IsOn = !led1IsOn;

} else if(req.indexOf("/led2") != -1){

setLedState(LED\_1, led2IsOn);

led2IsOn = !led2IsOn;

} else if(req.indexOf("/led3") != -1){

setLedState(LED\_2, led3IsOn);

led3IsOn = !led3IsOn;

}

else if (req.indexOf("/Up")!= -1){

myStepper.step(stepsPerRevolution);

}

else if (req.indexOf("/Down")!= -1)

{

myStepper.step(-stepsPerRevolution);

}

else{

Serial.println("invalid request");

client.stop();

return;

}

}

void setLedState(int led, bool state){

if(!state){

digitalWrite(led, HIGH);

} else {

digitalWrite(led, LOW);}

}

**ПРИЛОЖЕНИЕ Б**

package com.example.myapplication;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.SharedPreferences;  
import android.os.Bundle;  
import android.service.autofill.OnClickAction;  
import android.util.Log;  
import android.view.MotionEvent;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
  
import com.example.myapplication.databinding.ActivityMainBinding;  
  
import java.io.IOException;  
  
import okhttp3.OkHttpClient;  
import okhttp3.Request;  
import okhttp3.Response;  
import okhttp3.ResponseBody;  
  
public class MainActivity extends AppCompatActivity {  
 private Request request;  
 private ActivityMainBinding binding;  
 private SharedPreferences pref;  
 private OkHttpClient client = new OkHttpClient();  
 private EditText epsip;  
 private Button bSave;  
 private Button bLed1, bLed2, bLed3;  
 private ImageButton Up, Down;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 pref = getSharedPreferences("MyPref",MODE\_PRIVATE);  
  
  
 epsip = (EditText) findViewById(R.id.epsip);  
 bSave = (Button) findViewById(R.id.bSave);  
 bLed1 = (Button) findViewById(R.id.bLed1);  
 bLed2 = (Button) findViewById(R.id.bLed2);  
 bLed3 = (Button) findViewById(R.id.bLed3);  
 Up = (ImageButton) findViewById(R.id.UpButton);  
 Down = (ImageButton) findViewById(R.id.DownButton);  
 get\_ip();  
  
 View.OnClickListener onClickListener = new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 switch (view.getId())  
 {  
 case R.id.bLed1:  
 {  
 post("led1");  
 break;  
 }  
 case R.id.bLed2:  
 {  
 post("led2");  
 break;  
 }  
 case R.id.bLed3:  
 {  
 post("led3");  
 break;  
 }  
 }  
  
  
  
 }  
 };  
  
 View.OnTouchListener onTouchListener = new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View view, MotionEvent motionEvent) {  
 switch(view.getId())  
 {  
 case R.id.UpButton:  
 {  
 post("Up");  
 break;  
 }  
 case R.id.DownButton:  
 {  
 post("Down");  
 break;  
 }  
 }  
 return true;  
 }  
 };  
  
 Up.setOnTouchListener(onTouchListener);  
 Down.setOnTouchListener(onTouchListener);  
  
 bLed1.setOnClickListener(onClickListener);  
 bLed2.setOnClickListener(onClickListener);  
 bLed3.setOnClickListener(onClickListener);  
  
 bSave.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if(!epsip.getText().equals(null))  
 {  
 save\_ip(String.valueOf(epsip.getText()));  
 }  
 }  
 });  
  
  
 }  
  
 private void save\_ip(String ip)  
 {  
 SharedPreferences.Editor ed = pref.edit();  
 ed.putString("ip", ip);  
 ed.apply();  
 }  
  
 private void get\_ip()  
 {  
 String ip = pref.getString("ip", "");  
 if(ip!=null)  
 {  
 if(!ip.isEmpty())  
 {  
 epsip.setText(ip);  
 }  
 }  
 }  
  
 private void post(String post)  
 {  
 Thread t = new Thread(() -> {  
 request = new Request.Builder().url("http://"+epsip.getText()+"/"+post).build();  
 try{  
 Response response = client.newCall(request).execute();  
 if(response.isSuccessful())  
 {  
 String resultText = response.body().toString();  
 }  
  
 }  
 catch(IOException e)  
 {}  
 });  
 t.setDaemon(true);  
 t.start();  
 }  
}