МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пензенский государственный технологический университет»

(ПензГТУ)

Факультет информационных и образовательных технологий

Кафедра «Информационные технологии и системы»

Дисциплина «Программное и аппаратное обеспечение

информационных систем»

КОНТРОЛЬНАЯ РАБОТА

на тему: Разработка программно-аппаратного комплекса по мониторингу и автоматическому регулированию температуры и влажности в теплице.

Выполнили: студенты гр. 15ИС3м (группа №3)

Сивишкин А.О., Кирков А.О., Соломадина Е.А., Зелинская С.С., Глухова В., Приходько Р.В., Казакул А.С., Ясаревская Д.П.

Проверил: ст. преподаватель каф. ИТС Володин К.И.

Работа защищена с оценкой: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пенза, 2016

Содержание

[Введение 3](#_Toc452972610)

[Описание работы установки 4](#_Toc452972611)

[Компоненты установки 6](#_Toc452972612)

[Выводы 7](#_Toc452972613)

[Листинг программы для Arduino 8](#_Toc452972614)

[Листинг Android приложения 10](#_Toc452972615)

[Листинг интерфейса 15](#_Toc452972616)

[Список литературы 18](#_Toc452972617)

**Цель работы**: Разработать программно-аппаратный комплекс по мониторингу и автоматическому регулированию температуры и влажности в теплице.

**Задачи**:

1. собрать данные с датчиков;
2. передать данные на устройства с АРМ-процессором;
3. написать программу на языке c++, позволяющую производить обработку и визуализацию данных в системах на базе ARM-процессоров.

Введение

То, что не выходит из наших рук в течение всего дня – это смартфон. Современные мобильные телефоны – это уже не просто телефоны, это те устройства, которые во многом могут заменить и ноутбук, и настольный компьютер, тем самым экономя нам время и делая нашу жизнь проще и удобнее.

Современные смартфоны оснащены самыми передовыми технологиями: мощными процессорами, видеокартами, большими объемами оперативной памяти, экранами с высоким разрешением, функциями записи и воспроизведения звука студийного класса и многое другое. Если говорить о чипах для смартфонов, то на рынке безоговорочно доминирует архитектура ARM для семейства 32- и 64-битных процессоров, которая разрабатывается одноимённой компанией ARM Limited.

В 2007 году в 98% мобильных телефонах мира стоял как минимум один процессор ARM. Эти процессоры обладают низким энергопотреблением, именно поэтому широко распространены на рынке портативной техники и особенно в сфере мобильных устройств. У ARM много известных лицензиатов, которые делают собственные версии ядер на основе этой архитектуры: например, Qualcomm Shapdragon или Nvidia Tegra.

В настоящее время значимыми являются несколько семейств процессоров ARM:

* ARM7 (с тактовой частотой до 60-72 МГц), предназначенные, например, для недорогих мобильных телефонов и встраиваемых решений средней производительности. В настоящее время активно вытесняется новым семейством Cortex.
* ARM9 , ARM11 (с частотами до 1 ГГц) для более мощных телефонов, карманных компьютеров и встраиваемых решений высокой производительности.
* Cortex A — новое семейство процессоров на смену ARM9 и ARM11.
* Cortex M — новое семейство процессоров на смену ARM7, также призванное занять новую для ARM нишу встраиваемых решений низкой производительности. В семействе присутствуют четыре значимых ядра: Cortex M0, Cortex M3, Cortex M4 и Cortex-M7.

Описать крутые АРМ процессоры. Сделать акцент что в смартфоны сейчас устанавливают крутые АРМ процессоры.

Описание работы установки

Arduino сбирает данные с датчика влажности и температуры данные. Если температура повышается выше определённого лимита, то запускается сервопривод для закрытия окна. Данные о температуре, влажности и состоянии окна передаются через Bluetooth на приложение в сматфоне. В приложении отображаются текущие данные о температуре и влажности. По этим данным строятся динамические графики. Из мобильного приложения можно в ручном режиме закрыть или открыть окно.

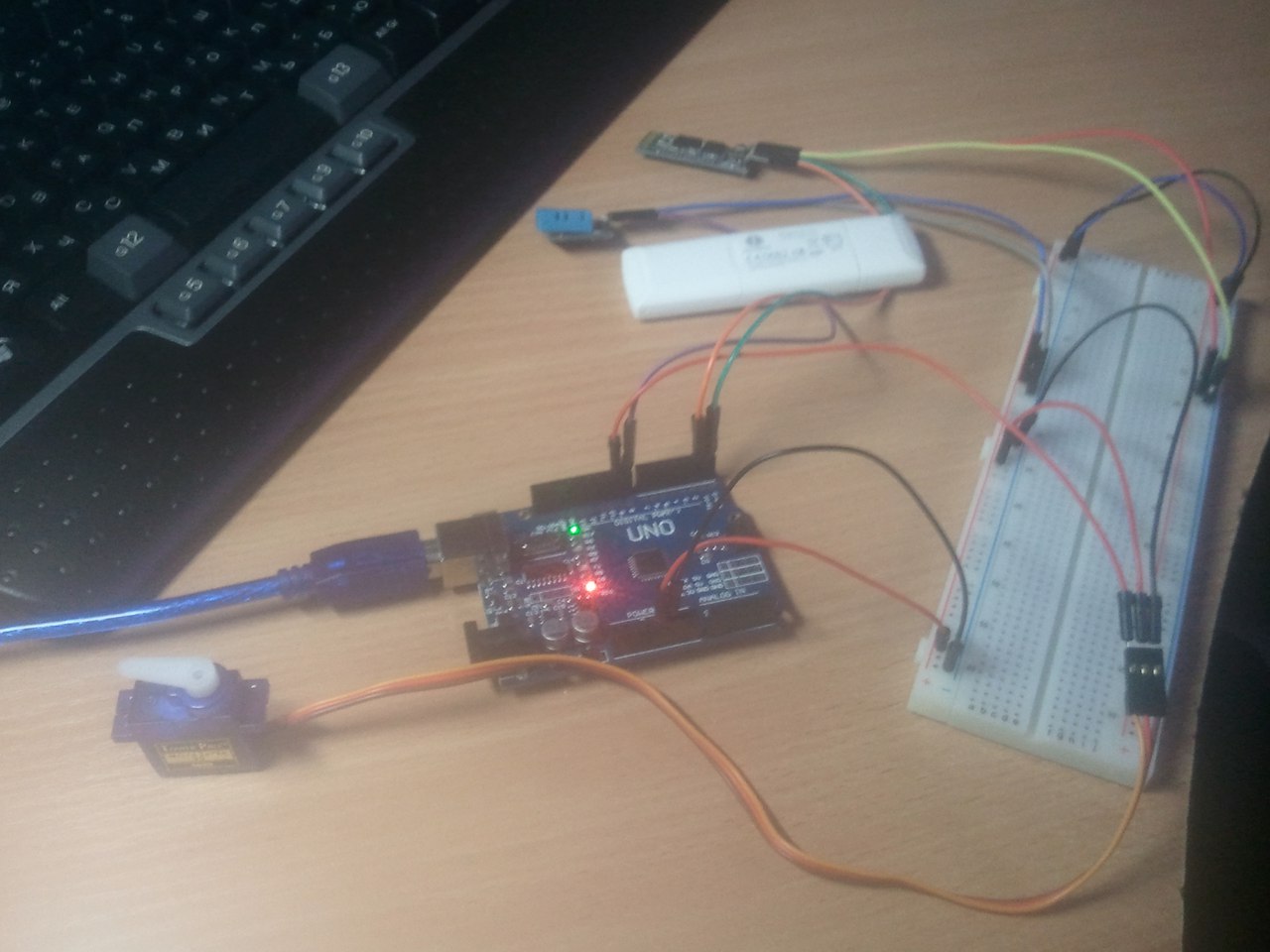


Рисунок 1 – Лабораторная установка

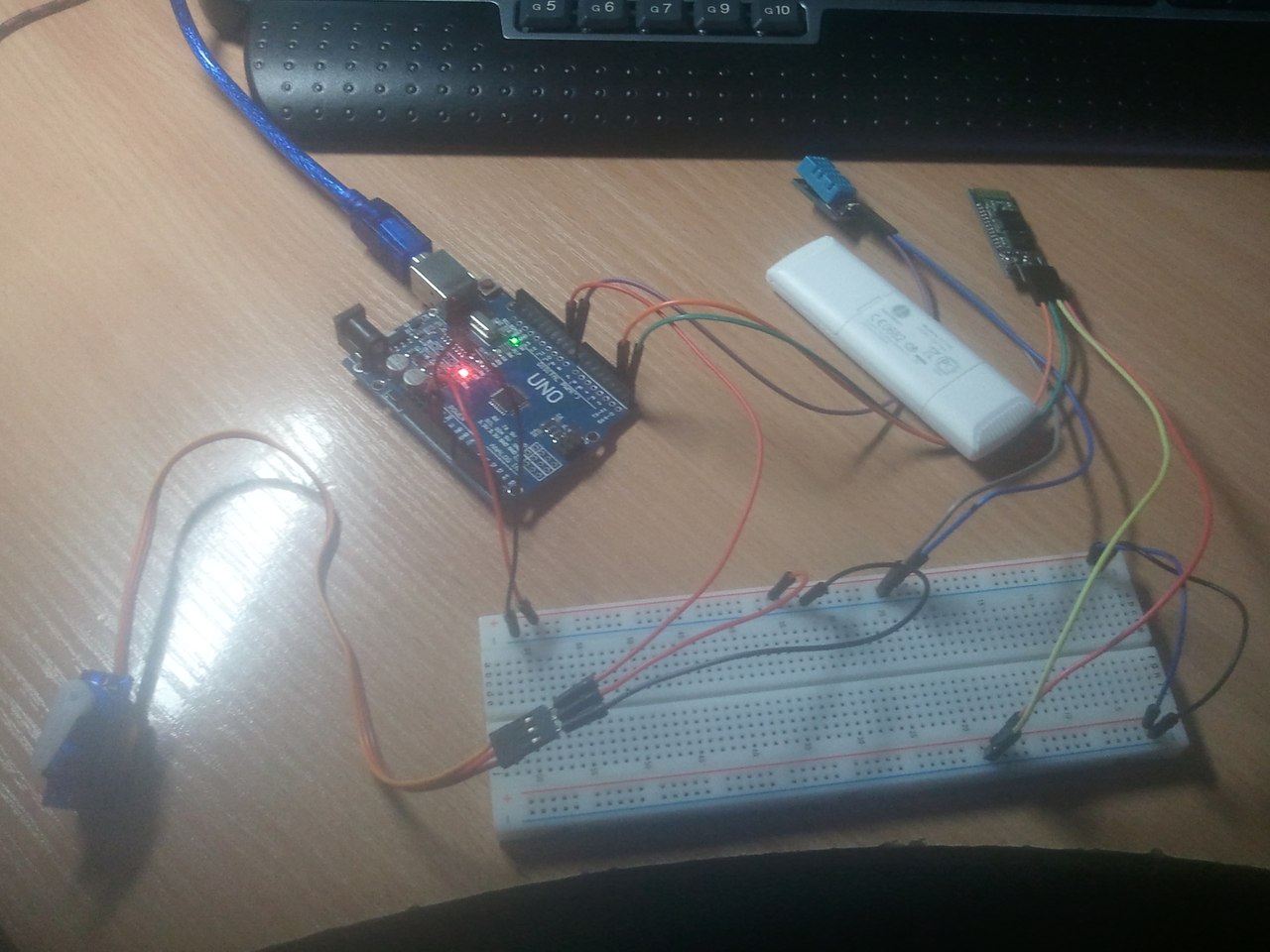


Рисунок 2 – Лабораторная установка

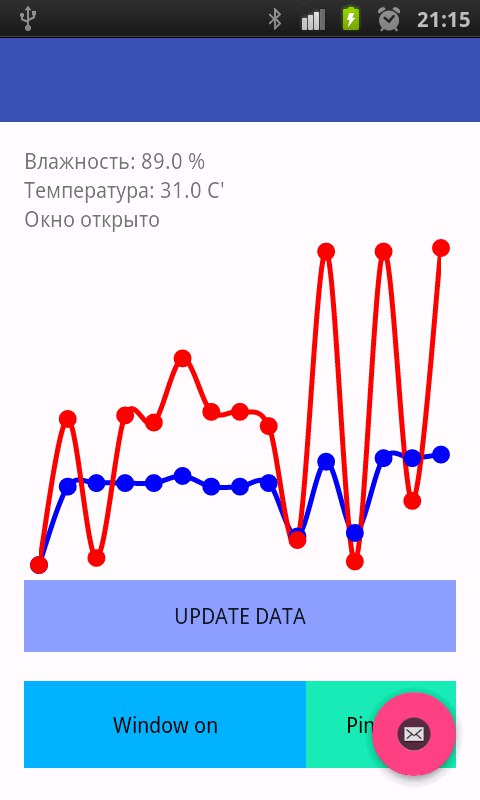


Рисунок 3 – Скриншот работающего приложения

Update data – делает запрос через bluetooth к arduino и обновляет данные о температуре и влажности.

Window on – открывает окно если оно закрыто, и закрывает если открыто.

Компоненты установки

|  |  |  |
| --- | --- | --- |
| № | Устройство | Назначение |
| 1 | Arduino Uno | Основная плата, которая собирает данные и передаёт на обрабатывающее устройство |
| 2 | DHT 11 | Датчик температуры и влажности |
| 3 | BT Module HC-05 | Блютуз модуль для беспроводного соединения |
| 4 | MicroServo sg90 | Серво привод для отрывания окна |
| 5 | Устройство с ОС Android | Устройство с АРМ-процессором, принимает данные |
| 6 | Мобильное приложение | Обрабатывает и отображает собранные данные. |

Выводы

Мы разработали програмно-аппаратного комплекс по мониторингу и автоматическому регулированию температуры и влажности в теплице.

Листинг программы для Arduino

#include "DHT.h"

#include <Servo.h> //используем библиотеку для работы с сервоприводом

#define DHTPIN 8 // номер пина, к которому подсоединен датчик

// Раскомментируйте в соответствии с используемым датчиком

// Инициируем датчик

Servo servo;

DHT dht(DHTPIN, DHT11);

float h =30.0;

float t =30.0;

boolean servoSatus = false;

boolean handOpen = false;

float tempLimit = 30.0;

void setup() {

//Устанавливаем скорость UART

Serial.begin(9600);

//Настраиваем нужные пины на выход

pinMode(6, OUTPUT);

pinMode(7, OUTPUT);

servo.attach(9);

servo.write(0);

//запуск дачика

dht.begin();

}

void loop() {

//Если данные пришли

if (Serial.available() > 0) {

Serial.print(h);

Serial.print(t);

if(servoSatus) {

Serial.print(1);

}

else{

Serial.print(0);

}

//Считываем пришедший байт

byte incomingByte = Serial.read();

if(incomingByte==75){

if(!servoSatus && !handOpen){

servo.write(180);

servoSatus=true;

handOpen= true;

}

else{

servo.write(0);

servoSatus=false;

handOpen= false;

}

}

//Получаем номер пина путем целочисленного деления значения принятого байта на 10

//и нужное нам действие за счет получения остатка от деления на 2:

//(1 - зажечь, 0 - погасить)

digitalWrite(incomingByte / 10, incomingByte % 2);

}

if(t>tempLimit && !handOpen){

if(!servoSatus){

servo.write(180);

}

servoSatus=true;

}

else if (!handOpen){

if(servoSatus){

servo.write(0);

}

servoSatus=false;

}

updateDate();

}

void updateDate(){

//Считываем влажность

float hh = dht.readHumidity();

// Считываем температуру

float tt = dht.readTemperature();

// Проверка удачно прошло ли считывание.

if (isnan(h) || isnan(t)) {

return;

}

t=tt;

h=hh;

}

Листинг Android приложения

**package** com.example.alexandr.tests3;  
  
**import** android.bluetooth.BluetoothAdapter;  
**import** android.bluetooth.BluetoothDevice;  
**import** android.bluetooth.BluetoothSocket;  
**import** android.content.Intent;  
**import** android.graphics.Color;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
**import** android.widget.ToggleButton;  
  
**import** org.apache.commons.io.IOUtils;  
  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.OutputStream;  
**import** java.lang.reflect.InvocationTargetException;  
**import** java.lang.reflect.Method;  
**import** java.util.ArrayList;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
**import** lecho.lib.hellocharts.gesture.ZoomType;  
**import** lecho.lib.hellocharts.model.Line;  
**import** lecho.lib.hellocharts.model.LineChartData;  
**import** lecho.lib.hellocharts.model.PointValue;  
**import** lecho.lib.hellocharts.view.LineChartView;  
  
**public class** MainActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 **private static** String *hemiVal* = **"0.00"**;  
 **private static** String *tempVal* = **"0.00"**;  
 **private static char** *winStatus* = **'1'**;  
 *//Экземпляры классов наших кнопок* TextView **tempV**;  
 TextView **hemiV**;  
 TextView **windV**;  
 Button **dataUpdateBtn**;  
 ToggleButton **pingBtn**;  
 ToggleButton **windowBtn**;  
 LineChartView **chart**;  
 List<Float> **tempCollection** = **new** LinkedList<>();  
 List<Float> **hemiCollection** = **new** LinkedList<>();  
 *//Сокет, с помощью которого мы будем отправлять данные на Arduino* BluetoothSocket **clientSocket**;  
  
 *//Эта функция запускается автоматически при запуске приложения* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **chart** = (LineChartView) findViewById(R.id.***chart***);  
  
 *//"Соединям" вид кнопки в окне приложения с реализацией* **dataUpdateBtn** = (Button) findViewById(R.id.***dataUpdate***);  
 **pingBtn** = (ToggleButton) findViewById(R.id.***ping***);  
 **windowBtn** = (ToggleButton) findViewById(R.id.***windowBtn***);  
  
 **tempV** = (TextView) findViewById(R.id.***temp***);  
 **hemiV** = (TextView) findViewById(R.id.***hemidy***);  
 **windV** = (TextView) findViewById(R.id.***window***);  
  
 *//Добавлем "слушатель нажатий" к кнопке* **dataUpdateBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 dataUpdate();  
 }  
 });  
  
 **pingBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 ping();  
 }  
 });  
 **windowBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 window();  
 }  
 });  
  
  
 btStart();  
 **new** ArduinoReader();  
 dataUpdate();  
 }  
  
 **private void** ping() {  
 *//Пытаемся послать данные* OutputStream outStream = **null**;  
 **try** {  
 *//Получаем выходной поток для передачи данных* outStream = **clientSocket**.getOutputStream();  
 *//Пишем данные в выходной поток* **int** value = (**pingBtn**.isChecked() ? 1 : 0) + 60;  
 outStream.write(value);  
  
 } **catch** (IOException e) {  
 *//Если есть ошибки, выводим их в лог* Log.*d*(**"BLUETOOTH"**, e.getMessage());  
 }  
 viewUpdate();  
 }  
  
 **private void** window() {  
 *//Пытаемся послать данные* OutputStream outStream = **null**;  
 **try** {  
 *//Получаем выходной поток для передачи данных* outStream = **clientSocket**.getOutputStream();  
 *//Пишем данные в выходной поток* outStream.write(75);  
  
 } **catch** (IOException e) {  
 *//Если есть ошибки, выводим их в лог* Log.*d*(**"BLUETOOTH"**, e.getMessage());  
 }  
 **windowBtn**.setChecked(!**windowBtn**.isChecked());  
 *winStatus* = *winStatus* == **'0'** ? **'1'** : **'0'**;  
 viewUpdate();  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu; this adds items to the action bar if it is present.* getMenuInflater().inflate(R.menu.***menu\_main***, menu);  
 **return true**;  
 }  
  
 **private void** btStart() {  
 *//Включаем bluetooth. Если он уже включен, то ничего не произойдет* String enableBT = BluetoothAdapter.***ACTION\_REQUEST\_ENABLE***;  
 startActivityForResult(**new** Intent(enableBT), 0);  
  
 *//Мы хотим использовать тот bluetooth-адаптер, который задается по умолчанию* BluetoothAdapter bluetooth = BluetoothAdapter.*getDefaultAdapter*();  
  
 *//Пытаемся проделать эти действия* **try** {  
 *//Устройство с данным адресом - наш Bluetooth Bee  
 //Адрес опредеяется следующим образом: установите соединение  
 //между ПК и модулем (пин: 1234), а затем посмотрите в настройках  
 //соединения адрес модуля. Скорее всего он будет аналогичным.* BluetoothDevice device = bluetooth.getRemoteDevice(**"30:14:10:24:14:33"**);  
  
 *//Инициируем соединение с устройством* Method m = device.getClass().getMethod(**"createRfcommSocket"**, **int**.**class**);  
  
 **clientSocket** = (BluetoothSocket) m.invoke(device, 1);  
 **clientSocket**.connect();  
  
 *//В случае появления любых ошибок, выводим в лог сообщение* } **catch** (IOException | SecurityException | IllegalArgumentException | NoSuchMethodException | InvocationTargetException | IllegalAccessException e) {  
 Log.*d*(**"BLUETOOTH"**, e.getMessage());  
 }  
  
 *//Выводим сообщение об успешном подключении* Toast.*makeText*(getApplicationContext(), **"CONNECTED"**, Toast.***LENGTH\_LONG***).show();  
 }  
  
 **private void** dataUpdate() {  
 **try** {  
 *//Получаем выходной поток для передачи данных* OutputStream outStream = **clientSocket**.getOutputStream();  
 outStream.write(60);  
 } **catch** (IOException e) {  
 *//Если есть ошибки, выводим их в лог* Log.*d*(**"BLUETOOTH"**, e.getMessage());  
 }  
 viewUpdate();  
 }  
  
 **private void** viewUpdate() {  
 **hemiCollection**.add(Float.*valueOf*(*hemiVal*.substring(0, 4)));  
 **tempCollection**.add(Float.*valueOf*(*tempVal*.substring(0, 4)));  
 **tempV**.setText(*tempVal*);  
 **hemiV**.setText(*hemiVal*);  
 **if** (*winStatus* == **'0'**) {  
 **windV**.setText(**"Окно закрыто"**);  
 **windowBtn**.setChecked(**false**);  
 } **else** {  
 **windV**.setText(**"Окно открыто"**);  
 **windowBtn**.setChecked(**true**);  
 }  
  
 updateChart();  
 }  
  
 **private void** updateChart() {  
 **if** (**tempCollection**.size() > 10 && **hemiCollection**.size() > 10) {  
 List<Float> temp = **tempCollection**;  
 List<Float> hemi = **hemiCollection**;  
 List<PointValue> valuesT = **new** ArrayList<>();  
 **for** (**int** i = 0; i < hemi.size(); i++) {  
 valuesT.add(**new** PointValue(i, hemi.get(i)));  
 }  
 List<PointValue> valuesH = **new** ArrayList<>();  
 **for** (**int** i = 0; i < temp.size(); i++) {  
 valuesH.add(**new** PointValue(i, temp.get(i)));  
 }  
  
 *//In most cased you can call data model methods in builder-pattern-like manner.* Line line1 = **new** Line(valuesH).setColor(Color.***BLUE***).setCubic(**true**);  
 Line line2 = **new** Line(valuesT).setColor(Color.***RED***).setCubic(**true**);  
 List<Line> lines = **new** ArrayList<>();  
 lines.add(line1);  
 lines.add(line2);  
  
 LineChartData data = **new** LineChartData(lines);  
 data.update(4);  
 **chart**.setZoomType(ZoomType.***HORIZONTAL\_AND\_VERTICAL***);  
 **chart**.setLineChartData(data);  
 }  
  
 }  
  
 @Override  
 **public void** onClick(View v) {  
  
 }  
  
 **private class** ArduinoReader **implements** Runnable {  
  
 **public** ArduinoReader() {  
 **new** Thread(**this**).start();  
 }  
  
 **public void** run() {  
 **byte**[] buffer = **new byte**[256]; *// buffer store for the stream* **int** bytes; *// bytes returned from read()* InputStream tmpIn = **null**;  
  
 **try** {  
 tmpIn = **clientSocket**.getInputStream();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 *// Keep listening to the InputStream until an exception occurs* **while** (**true**) {  
 **try** {  
 *// Read from the InputStream* **assert** tmpIn != **null**;  
 bytes = tmpIn.read(buffer);  
 String response = IOUtils.*toString*(buffer, **"UTF-8"**);  
 System.***out***.println(response);  
  
 *hemiVal* = response.substring(0, 4) + **" %"**;  
 *tempVal* = response.substring(5, 9) + **" C'"**;  
 *winStatus* = response.charAt(10);  
  
 } **catch** (IOException e) {  
 **try** {  
 tmpIn.close();  
 } **catch** (IOException ee) {  
 ee.printStackTrace();  
 }  
 **break**;  
 }  
 }  
 }  
 }  
}

Листинг интерфейса

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:context="com.example.alexandr.tests3.MainActivity"  
 tools:showIn="@layout/activity\_main"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/label1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Влажность: "** />  
  
 <**TextView  
 android:id="@+id/hemidy"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0.00%"** />  
 </**LinearLayout**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/label2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Температура: "** />  
  
 <**TextView  
 android:id="@+id/temp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0 C\*"** />  
 </**LinearLayout**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/window"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Cостояние окна"** />  
 </**LinearLayout**>  
  
 <**lecho.lib.hellocharts.view.LineChartView  
 android:id="@+id/chart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="231dp"** />  
  
 <**Button  
 android:id="@+id/dataUpdate"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#8c9eff"  
 android:backgroundTint="#c28d9eff"  
 android:clickable="false"  
 android:enabled="true"  
 android:text="Update data"** />  
  
 <**TextView  
 android:id="@+id/label3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"** />  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:weightSum="1"**>  
  
 <**ToggleButton  
 android:id="@+id/windowBtn"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="188dp"  
 android:layout\_height="58dp"  
 android:background="#00b0ff"  
 android:clickable="false"  
 android:enabled="true"  
 android:text="Window"  
 android:textOff="Window off"  
 android:textOn="Window on"** />  
  
 <**ToggleButton  
 android:id="@+id/ping"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="match\_parent"  
 android:layout\_height="58dp"  
 android:background="#1de9b6"  
 android:clickable="false"  
 android:enabled="true"  
 android:text="Ping"  
 android:textOff="Ping off"  
 android:textOn="Ping on"  
 android:layout\_gravity="right"** />  
  
 </**LinearLayout**>  
  
 </**LinearLayout**>  
  
  
 </**LinearLayout**>  
  
  
</**RelativeLayout**>

Список литературы

* 1. Подключение датчика температуры и влажности DHT11 и DHT22.: [Электронный ресурс] // Занимательная робототехника: [сайт]. [2016]. URL: http://edurobots.ru/2015/02/arduino-dlya-nachinayushhix-urok-9-podklyuchenie-datchika-temperatury-i-vlazhnosti-dht11-i-dht22/, (дата обращения 14.04.2016).
  2. Redesign DHT11 Temperature and Relative Humidity Sensor Module : [Электронный ресурс] // ChiOSZ robots: [сайт]. [2016]. URL: http://chioszrobots.com/2013/07/15/redesign-dht11-temperature-and-relative-humidity-sensor-module, (дата обращения 14.04.2016).
  3. Управление сервоприводом : [Электронный ресурс] // Занимательная робототехника: [сайт]. [2016]. URL: http://edurobots.ru/2014/04/arduino-servoprivod/, (дата обращения 14.04.2016).
  4. Servo : [Электронный ресурс] //Arduino: [сайт]. [2016]. URL: https://www.arduino.cc/en/Reference/ServoRead, (дата обращения 14.04.2016).