2

Міністерство освіти та науки України

Національний технічний університет України «КПІ» ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

**Курсовий робота**

з дисципліни «Проектування мікропроцесорних систем»

на тему

«Bluetooth-керуємий іграшковий автомобіль»

|  |  |
| --- | --- |
| **Керівник:** доц. Долина В. Г.  «Захист дозволено»  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 р.  Захищено з оцінкою  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **/** підпис керівника / | **Виконав:**  студент IV курсу  групи ЗПІ-зп61  Макіян С.А (№ ЗПІ-зп6115) |

**Київ – 2019**

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

**КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ У ТЕХНІЧНИХ СИСТЕМАХ**

Дисципліна Проектування мікропроцесорних систем

СпеціальністьІнженерія програмного забезпечення

Курс 4 Група ЗПІ-зп61 Семестр V

**ЗАВДАННЯ**

На курсовий проект (роботу) студента

Макіяна Смбата Артуровича

1. **Тема проекту (роботи): «Bluetooth-керуємий іграшковий автомобіль»**
2. **Термін здачі студентом завершеного проекту (роботи) :** 16.01.2019.
3. **Вхідні дані до проекту (роботи):** c++*,Java*
4. **Зміст розрахунково–пояснювальної записки (перелік питань, які підлягають розробці):**
   1. *Постановка задачі;*
   2. *Ієрархія задач, що вирішуються;*
   3. *Аналіз функціоналу;*
   4. *Інструкція користувачу;*
   5. *Інструкція програмісту;*
   6. *Графічні матеріали.*
5. **Дата отримання завдання:** 02.11.2018.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапу курсового проекту (роботи) | Строк виконання етапів проекту (роботи) | Примітки |
| 1 | Отримання завдання для курсового проект | 02.11.2018 |  |
| 2 | Вивчення предметної області | 10.11.2018 |  |
| 3 | Проектування архітектури модуля | 20.11.2018 |  |
| 4 | Розробка програмного продукту | 10.12.2018 |  |
| 5 | Проектування та розробка інтерфейсу | 20.12.2018 |  |
| 6 | Налагодження та тестування ПП | 01.01.2019 |  |
| 7 | Виправлення виявлених помилок та недоліків | 5.01.2019 |  |
| 8 | Оформлення документації | 25.05.2018 |  |
| 9 | Здача курсового проекту | 07.06.2018 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** |  |  | Макіян Смбат Артурович |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник** |  |  | Долина Віктор Георгійович |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 р

Зміст

[ВСТУП 5](#_Toc535390651)

[1. ПОСТАНОВКА ЗАДАЧІ 6](#_Toc535390652)

[2. МАТЕМАТИЧНА МОДЕЛЬ 7](#_Toc535390653)

[3. ВИБІР ПРОГРАМНОГО СЕРЕДОВИЩА 11](#_Toc535390654)

[3.1 Опис та обґрунтування вибору програмного середовища плати 11](#_Toc535390655)

[4 Алгоритм роботи коду мікроконтролера 12](#_Toc535390656)

[**Висновки** 13](#_Toc535390657)

[Перелік використаних джерел 14](#_Toc535390658)

[Додатки 15](#_Toc535390659)

## ВСТУП

«Іграшка «Радіокеруємий автомобіль»»

*Актуальність теми*: більшість людства для доступу до інформації використовує власний смартфон, тож вирішено було перепроектувати радіокеруюємий іграшковий автомобіль, під керування зі смартфону через Bluetooth інтерфейс.

*Метою курсової роботи* є створення програмного та апаратного комплексу, котрий дозволить керувати іграшковим автомобілем зі смартфону.

Завдання курсової роботи обумовлене її метою:

* переробити іграшковий автомобіль під керівництво через Bluetooth за допомогою мікроконтроллера.
* написати програмне забезпечення для смартфону для керування іграшкою через Bluetooth.

# ПОСТАНОВКА ЗАДАЧІ

**Розробити апаратно програмний комплекс для керування іграшковою машинкою зі смартфона:**

**Що до апаратної частини:**

* **в якості шасі використати існуючу іграшку;**
* **нова керуюча мікросхема з обв’язкою має бути компактною щоби поміститися в старий корпус іграшки;**
* **мова прошивки контролеру має бути достатньо простою, що полегшить його підтримку.**

**Програмна частина:**

* **необхідно написати мобільний додаток під операційну систему Android для керування іграшкою через Bluetooth.**

**В якості мікроконтролеру було вирішино використати Arduino Nano, в зв’язці з Bluetooth shield HC-06, в якості драйверу двигунів L239D.**

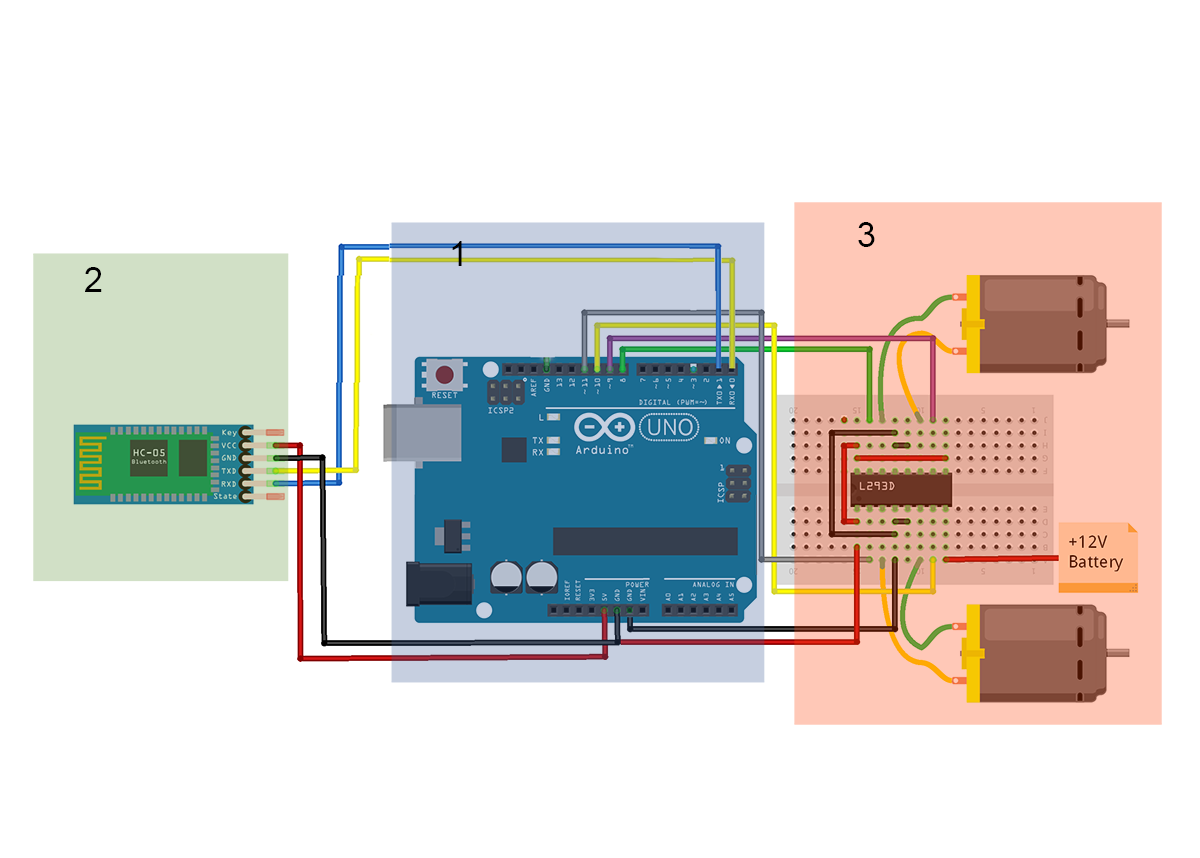
**Переваги такої схеми:**

* **дешеві та компактні компоненти;**
* **велике ком’юніті та підтримка;**
* **маленький поріг входу.**

# 2. МАТЕМАТИЧНА МОДЕЛЬ

Апаратну частину іграшки легко представити як комплекс з 3 компонентів логічних компонентів:

1. Керуюча плата Adruino Nano
2. Модуль Bluetooth HC-06
3. Силова частина (драйвер+двигуни)



**2.1 Логічні блоки апаратної частини**

**Керуюча плата Arduino Nano:**

Платформа Nano, побудована на мікроконтролері ATmega328 (Arduino Nano 3.0), має невеликі розміри і може використовуватися для прототипів вбудованих систем. Nano розроблена і продається компанією Gravitech.

**Короткі характеристики**

|  |  |
| --- | --- |
| Мікроконтролер | ATmega328 |
| Робоча напруга (логічна рівень) | 5 В |
| Вхідна напруга (рекомендований) | 7-12 В |
| Вхідна напруга (граничне) | 6-20 В |
| Цифрові Входи / Виходи | 14 (6 з яких можуть використовуватися як виходи ШІМ) |
| аналогові входи | 8 |
| Постійний струм через вхід / вихід | 40 мА |
| Флеш пам `ять | 16 Кб (ATmega168) або 32 Кб (ATmega328) при цьому 2 КБ використовуються для завантажувача |
| ОЗУ | 1 Кб (ATmega168) або 2 КБ (ATmega328) |
| EEPROM | 1 Кб (ATmega328) |
| Тактова частота | 16 МГц |
| Розміри | 1.85 см x 4.2 см |

**Живлення:**

Arduino Nano може отримувати харчування через підключення Mini-B USB, або від нерегульованого 6-20 В (вихід 30), або регульованого 5 В (вихід 27), зовнішнього джерела живлення. Автоматично вибирається джерело з найвищим напругою.

Мікросхема FTDI FT232RL отримує харчування, тільки якщо сама платформа запитана від USB. Таким чином при роботі від зовнішнього джерела (НЕ USB), буде відсутня напруга 3.3 В, що генерується мікросхемою FTDI, при цьому світлодіоди RX і TX блимають тільки при наявності сигналу високого рівня на висновках 0 і 1.

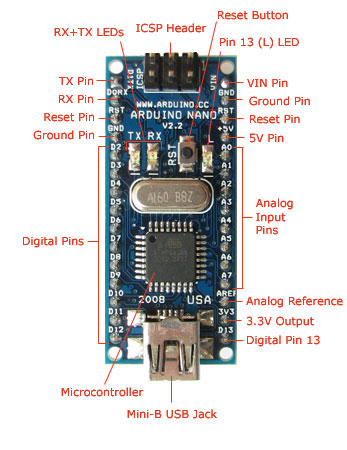
**Пам'ять**

Мікроконтролер ATmega328, має 32 кБ (в обох випадках 2 кБ икористовується для зберігання завантажувача). ATmega168 має 1 кБ ОЗУ і 512 байт EEPROM (яка читається і записується за допомогою бібліотеки EEPROM ), а ATmega328 - 2 кБ ОЗУ і 1 Кб EEPROM.

**Входи та вихіди**

Кожен з 14 цифрових висновків Nano, використовуючи функції pinMode() , digitalWrite() , і digitalRead () , може налаштовуватися як вхід або вихід. Висновки працюють при напрузі 5 В. Кожен висновок має навантажувальний резистор (стандартно відключений) 20-50 кОм і може пропускати до 40 мА.

Входи та віходи плати можно подивитися на рисунку 2.2

****

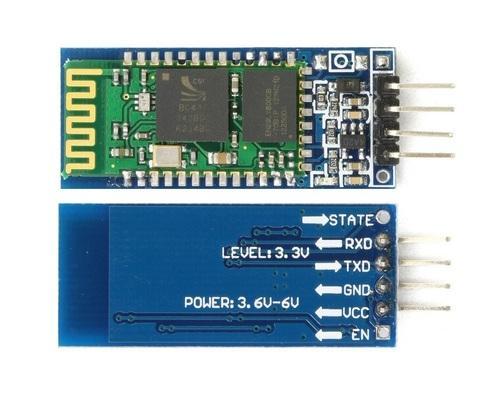
**2.2 Плата Arduino Nano**

**Послідовна шина: 0 (RX) і 1 (TX).** Висновки використовуються для отримання (RX) і передачі (TX) даних TTL. Дані висновки підключені до відповідних висновків мікросхеми послідовної шини FTDI USB-to-TTL.

**ШІМ: 3, 5, 6, 9, 10, і 11.** Будь-який з висновків забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції analogWrite ().

**Плата HC-06**

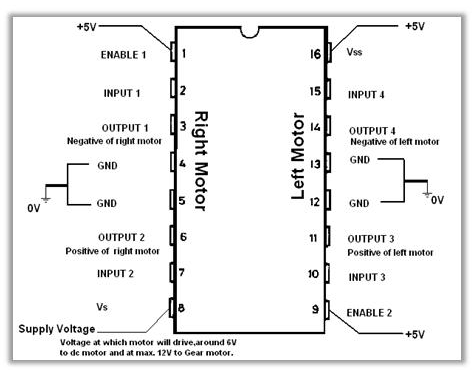
Проста в використанні плата котра реалізує взаємодію інтерфейсів Bluetooth та UART. Живиться від контакті Arduino Vcc 3.3, Gnd, та підключається до неї Rx->Tx, Tx->Rx. Плата показана на рис. 2.3



**2.3 Плата HC-06**

**Драйвер двигунів L239D**

Простий в використанні драйвер двигунів під Arduino. Дозволяє одночасно керувати 2 двигунами. Схематичне зображення плати дивись на рис 2.4

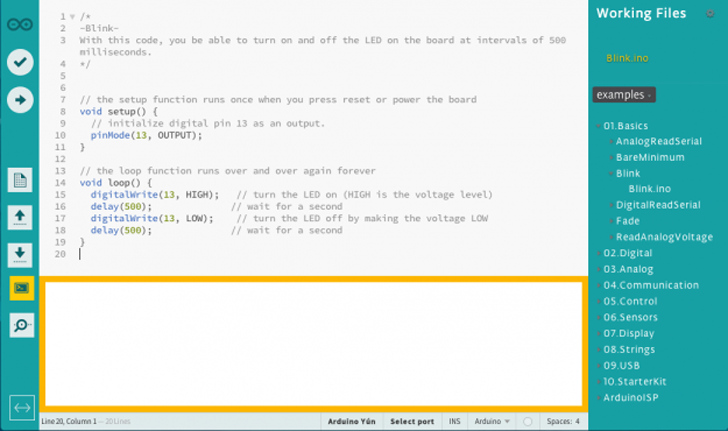


**2.3 Драйвер L239D**

# ВИБІР ПРОГРАМНОГО СЕРЕДОВИЩА

## 3.1 Опис та обґрунтування вибору програмного середовища плати

Для написання прошивки, доцільно використати Arduino Studio, котра є стандартною програмою для написання та заливки прошивки для плати Arduino. Мовою програмування за замоченням є діалект мови с++, адаптований під Arduino.



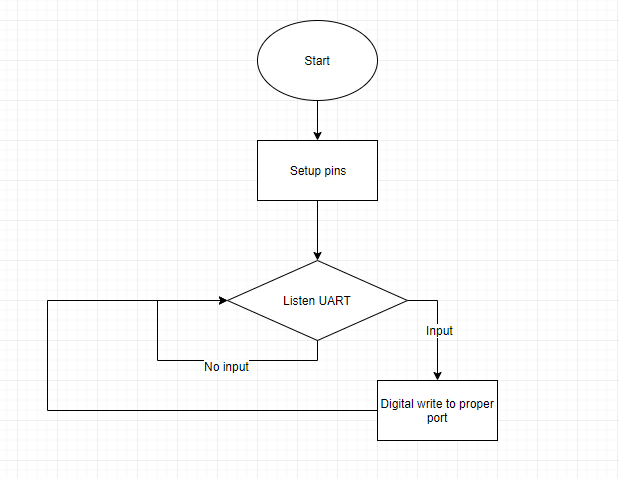
**3.1 Arduino Studio**

# Алгоритм роботи коду мікроконтролера

Мікроконтролер працює по наступній схемі:

* Спочатку виставляємо порти на запис та прослуховування;
* Після цього якщо на порт подається сигнал то, подється напруга на керуючий pin;
* Якщо на UART (Rx/Tx) не подється значення, то сигнали зі всіх портів віключаються.

Логічне зображення роботи мікроконтролера див. на рис. 4.1

****

**4.1 Блок схема логіки Arduino**

**Логіка роботи мобільного додатку.**

Логіка роботи мобільного наступна: якщо користувач керує віртуальним джойстиком, то треба віправити на Bluetooth adapter підключений до контролера код з напрямком, якщо ні, то нічого не відправляти.

Детальний код роботи контролера та клієнта дивись в додатку А та Б.

# ВИСНОВКИ

В ході виконання даного курсового проекту була спроектована система з мікрокотролеру та периферії, котра реалізує керування іграшковою машинкоє через інтерфейс Bluetooth. Хоча рішення працює, біло виявлено декілька недоліків такої системи:

* Інтерфейс Bluetooth дуже повільний;
* Розпайка на макетці та самі провода дуже не надійні;
* Дешева плата HC-06 при слабкому живленні (3x AA 1.5 В батарейки), постійно вірубається, тому живлення було змінено на дві батареї 3.7В 14500.

# Перелік використаних джерел

* 1. <http://arduino.ru/Hardware/ArduinoBoardNano>
  2. <https://www.arduino.cc/en/Guide/ArduinoNano>
  3. <https://www.indiamart.com/proddetail/bluetooth-hc06-serial-wireless-rf-transceiver-module-18007754791.html>
  4. <http://arduinoguides.blogspot.com/2012/06/using-l239-motor-driver.html>

## Додатки

**Додаток А**

**Arduino Sketch**

// from 5 to 9

void setup() {

Serial.begin(9600);

for (int i =5; i<=9; i++)

pinMode (i, OUTPUT);

}

void loop() {

if (Serial.available() > 0) {

byte incomingByte = Serial.read();

for (int i =5; i<=9; i++)

digitalWrite (i, LOW);

for (int i =5; i<=9-5+incomingByte; ++i)

digitalWrite(i, HIGH);

}

}

**Додаток Б**

**Код мобыльного додатку**

**Class MainActivity.java**

package software.idis.rc\_porsche;

import java.io.IOException;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

public class MainActivity extends Activity {

private Button bt, retry;

public static BluetoothSocket clientSocket;

public volatile static boolean masterExit = false;

private BluetoothAdapter bluetooth;

@Override

protected void onCreate(Bundle savedInstanceState) {

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

retry = (Button)findViewById(R.id.btn\_try);

bt = (Button)findViewById(R.id.button1);

retry.setText(getString(R.string.retry));

bt.setText(getString(R.string.bl\_not\_connected));

bt.setClickable(false);

tryToConnect();

}

private void tryToConnect()

{

String enableBT = BluetoothAdapter.ACTION\_REQUEST\_ENABLE;

startActivityForResult(new Intent(enableBT), 0);

bluetooth = BluetoothAdapter.getDefaultAdapter();

try{

BluetoothDevice device = bluetooth.getRemoteDevice("30:14:09:03:00:72");

Method m = device.getClass().getMethod(

"createRfcommSocket", new Class[] {int.class});

clientSocket = (BluetoothSocket) m.invoke(device, 1);

clientSocket.connect();

Toast.makeText(getApplicationContext(), "CONNECTED", Toast.LENGTH\_LONG).show();

} catch (IOException e) {

Log.d("BLUETOOTH", e.getMessage());

} catch (SecurityException e) {

Log.d("BLUETOOTH", e.getMessage());

} catch (NoSuchMethodException e) {

Log.d("BLUETOOTH", e.getMessage());

} catch (IllegalArgumentException e) {

Log.d("BLUETOOTH", e.getMessage());

} catch (IllegalAccessException e) {

Log.d("BLUETOOTH", e.getMessage());

} catch (InvocationTargetException e) {

Log.d("BLUETOOTH", e.getMessage());

}

if (clientSocket.isConnected())

{

bt.setText(getString(R.string.start\_rc));

bt.setClickable(true);

}

}

public void onClickRetry(View v)

{

tryToConnect();

}

public void onClickStart (View v)

{

Intent i = new Intent (this, ControlActivity.class);

startActivity(i);

}

public void onClickExit (View v)

{

masterExit = true;

this.finish();

}

}

**Class ControlView.java**

package software.idis.rc\_porsche;

import software.idis.rc\_porsche.TrianglePath;

import java.util.TreeMap;

import java.util.concurrent.atomic.AtomicInteger;

import android.annotation.SuppressLint;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Point;

import android.graphics.PointF;

import android.util.DisplayMetrics;

import android.util.SparseArray;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

public class ControlsView extends SurfaceView {

public static DisplayMetrics displayMetrics;

private int displayWidth, displayHeight;

private int circleX, circleY, circleRadius;

private Paint circleDefault, circleTouched, textPaint, triangleDefault, triangleTouched;

private TrianglePath up, down, right, left;

private TreeMap <String, TrianglePath> navHolder;

private TreeMap <String, Boolean> navHolderState;

private boolean lightCurrentTouched = false, lightPreviousTouched = false, lightOn = false;

private final String ST\_UP = "up", ST\_DOWN = "down", ST\_LEFT = "left", ST\_RIGHT = "right", ST\_EMPTY = "";

private String upDown = ST\_EMPTY;

private String leftRight = ST\_EMPTY;

static public AtomicInteger currentSendvalue;

/\*\* holder for touched points \*/

private SparseArray<PointF> mActivePointers;

/\*\* Remote loop class object GameLoopThread\*/

private GameThread mThread;

/\*\* Variable that lunches draw thread\*/

private boolean running = false;

//-------------Start of GameThread--------------------------------------------------\\

public class GameThread extends Thread

{

/\*\* Class object \*/

private ControlsView view;

/\*\* Constructor \*/

public GameThread(ControlsView view)

{

this.view = view;

}

/\*\* Set thread state \*/

public void setRunning(boolean run)

{

running = run;

}

/\*\* Thread actions \*/

@SuppressLint("WrongCall")

public void run()

{

while (running)

{

Canvas canvas = null;

try

{

// preparing Canvas

canvas = view.getHolder().lockCanvas();

synchronized (view.getHolder())

{

// Draw

onDraw(canvas);

//draw(canvas);

}

}

catch (Exception e) { }

finally

{

if (canvas != null)

{

view.getHolder().unlockCanvasAndPost(canvas);

}

}

}

}

}

//-------------End of GameThread--------------------------------------------------\\

public ControlsView(Context context)

{

super(context);

currentSendvalue = new AtomicInteger();

currentSendvalue.set(0);

mActivePointers = new SparseArray<PointF>();

mThread = new GameThread(this);

displayWidth = displayMetrics.widthPixels;

displayHeight = displayMetrics.heightPixels;

circleDefault = new Paint();

circleDefault.setColor(0xff00ff00);

circleTouched = new Paint();

circleTouched.setColor(0xffff0000);

textPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

textPaint.setTextSize(20);

triangleDefault = new Paint();

triangleDefault.setStrokeWidth(4);

triangleDefault.setColor(0xff0000ff);

triangleDefault.setStyle(Paint.Style.FILL\_AND\_STROKE);

triangleDefault.setAntiAlias(true);

triangleTouched = new Paint();

triangleTouched.setStrokeWidth(4);

triangleTouched.setColor(0xffffa500);

triangleTouched.setStyle(Paint.Style.FILL\_AND\_STROKE);

triangleTouched.setAntiAlias(true);

int heightCorrect = displayHeight/6;

int sideSize = displayHeight/4;

int sideOffset = displayWidth/7;

int initialPoint = displayWidth/35;

int marging = displayWidth /20;

Point a = new Point(initialPoint, displayHeight/2 + heightCorrect);

Point b = new Point(initialPoint + sideOffset, displayHeight/2 + sideSize + heightCorrect);

Point c = new Point(initialPoint + sideOffset, displayHeight/2 - sideSize + heightCorrect);

left = new TrianglePath(a, b, c);

initialPoint = initialPoint + sideOffset + marging;

a = new Point(initialPoint + sideOffset, displayHeight/2 + heightCorrect);

b = new Point(initialPoint , displayHeight/2 + sideSize + heightCorrect);

c = new Point(initialPoint , displayHeight/2 - sideSize + heightCorrect);

right = new TrianglePath(a, b, c);

initialPoint = displayWidth \* 3 / 4 + marging;

a = new Point(initialPoint , displayHeight/2 - sideOffset - marging /2 + heightCorrect);

b = new Point(initialPoint - sideSize, displayHeight/2 - marging /2 + heightCorrect);

c = new Point(initialPoint + sideSize, displayHeight/2 - marging /2 + heightCorrect);

up = new TrianglePath(a, b, c);

a = new Point(initialPoint , displayHeight/2 + sideOffset + marging /2 + heightCorrect);

b = new Point(initialPoint - sideSize, displayHeight/2 + marging /2 + heightCorrect);

c = new Point(initialPoint + sideSize, displayHeight/2 + marging /2 + heightCorrect);

down = new TrianglePath(a, b, c);

navHolder = new TreeMap<String, TrianglePath>();

navHolder.put(ST\_UP, up);

navHolder.put(ST\_DOWN, down);

navHolder.put(ST\_RIGHT, right);

navHolder.put(ST\_LEFT, left);

navHolderState = new TreeMap<String, Boolean>();

navHolderState.put(ST\_UP, false);

navHolderState.put(ST\_DOWN, false);

navHolderState.put(ST\_RIGHT, false);

navHolderState.put(ST\_LEFT, false);

circleX = displayWidth/2;

circleY = displayHeight/4;

circleRadius = displayHeight/10;

/\*\* Drawing everything \*/

getHolder().addCallback(new SurfaceHolder.Callback()

{

/\*\*\* Destroy draw surface \*/

public void surfaceDestroyed(SurfaceHolder holder)

{

boolean retry = true;

mThread.setRunning(false);

while (retry)

{

try

{

// waiting end of the tread

mThread.join();

retry = false;

}

catch (InterruptedException e) { }

}

}

/\*\* Creating draw surface \*/

public void surfaceCreated(SurfaceHolder holder)

{

mThread.setRunning(true);

mThread.start();

}

/\*\* If surface changed \*/

public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)

{

}

});

}

/\*\* Function that drawing sprites and background\*/

@SuppressLint("WrongCall")

protected void onDraw(Canvas canvas) {

canvas.drawColor(Color.WHITE);

canvas.drawText("Stub text: " , 10, 40 , textPaint);

canvas.drawCircle(circleX, circleY, circleRadius, circleDefault);

for (String key: navHolder.keySet())

{

canvas.drawPath(navHolder.get(key).getPath(), triangleDefault);

}

for (String key: navHolder.keySet())

{

navHolderState.put(key, false);

}

lightCurrentTouched = false;

for (int size = mActivePointers.size(), i = 0; i < size; i++) {

PointF point = mActivePointers.valueAt(i);

if (point != null)

{

if (IsCircleTouched(point, circleX, circleY, circleRadius))

lightCurrentTouched = true;

//check is triangle touched

for (String key: navHolder.keySet())

{

navHolderState.put(key, IsTriangleTouched(point, navHolder.get(key)) || navHolderState.get(key));

}

}

}

if (lightCurrentTouched == false && lightPreviousTouched == true)

{

lightOn = !lightOn;

}

lightPreviousTouched = lightCurrentTouched;

if (lightOn == true)

canvas.drawCircle(circleX, circleY, circleRadius, circleTouched);

//decide direction up down

if ( navHolderState.get(ST\_UP) == true || navHolderState.get(ST\_DOWN) == true )

{

boolean upTouched = navHolderState.get(ST\_UP);

boolean downTouched = navHolderState.get(ST\_DOWN);

if (upTouched == true && downTouched == true)

{

if ( upDown == ST\_EMPTY )

upDown = ST\_UP;

}

else if (upTouched == true)

upDown = ST\_UP;

else

upDown = ST\_DOWN;

}

else

upDown = ST\_EMPTY;

//decide direction right left

if ( navHolderState.get(ST\_LEFT) == true || navHolderState.get(ST\_RIGHT) == true )

{

boolean leftTouched = navHolderState.get(ST\_LEFT);

boolean rightTouched = navHolderState.get(ST\_RIGHT);

if (leftTouched == true && rightTouched == true)

{

if ( leftRight == ST\_EMPTY )

leftRight = ST\_LEFT;

}

else if (leftTouched == true)

leftRight = ST\_LEFT;

else

leftRight = ST\_RIGHT;

}

else

leftRight = ST\_EMPTY;

int sendResult = 0;

if (leftRight != ST\_EMPTY )

{

canvas.drawPath(navHolder.get(leftRight).getPath(), triangleTouched);

sendResult = sendResult | 1;

if (leftRight == ST\_LEFT)

sendResult = sendResult | 1<<1;

}

if (upDown != ST\_EMPTY)

{

canvas.drawPath(navHolder.get(upDown).getPath(), triangleTouched);

sendResult = sendResult | 1<<2;

if (upDown == ST\_UP)

sendResult = sendResult | 1<<3;

}

if (lightOn)

sendResult = sendResult | 1<<4;

currentSendvalue.set(sendResult);

}

@Override

public boolean onTouchEvent(MotionEvent event) {

// get pointer index from the event object

int pointerIndex = event.getActionIndex();

// get pointer ID

int pointerId = event.getPointerId(pointerIndex);

// get masked (not specific to a pointer) action

int maskedAction = event.getActionMasked();

switch (maskedAction) {

case MotionEvent.ACTION\_DOWN:

case MotionEvent.ACTION\_POINTER\_DOWN: {

// We have a new pointer. Lets add it to the list of pointers

// three points can be used same time

PointF f = new PointF();

f.x = event.getX(pointerIndex);

f.y = event.getY(pointerIndex);

mActivePointers.put(pointerId, f);

break;

}

case MotionEvent.ACTION\_MOVE: { // a pointer was moved

for (int size = event.getPointerCount(), i = 0; i < size; i++) {

PointF point = mActivePointers.get(event.getPointerId(i));

if (point != null) {

point.x = event.getX(i);

point.y = event.getY(i);

}

}

break;

}

case MotionEvent.ACTION\_UP:

case MotionEvent.ACTION\_POINTER\_UP:

case MotionEvent.ACTION\_CANCEL: {

mActivePointers.remove(pointerId);

break;

}

}

invalidate();

return true;

}

@Override

public boolean performClick() {

// TODO Auto-generated method stub

return super.performClick();

}

private boolean IsCircleTouched(PointF oneTouch, int x, int y, int radius)

{

double distance = Math.sqrt(Math.pow((oneTouch.x - x),2) + Math.pow((oneTouch.y - y),2) );

if (radius < distance)

return false;

else

return true;

}

private boolean IsTriangleTouched (PointF oneTouch, TrianglePath tr)

{

boolean b1, b2, b3;

b1 = sign(oneTouch, tr.a, tr.b) < 0.0f;

b2 = sign(oneTouch, tr.b, tr.c) < 0.0f;

b3 = sign(oneTouch, tr.c, tr.a) < 0.0f;

return ((b1 == b2) && (b2 == b3));

}

private float sign(PointF v1, Point v2, Point v3)

{

return (v1.x - v3.x) \* (v2.y - v3.y) - (v2.x - v3.x) \* (v1.y - v3.y);

}

}

**Class ControlActivity.java**

package software.idis.rc\_porsche;

import java.io.IOException;

import java.io.OutputStream;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.Intent;

import android.content.pm.ActivityInfo;

import android.os.Bundle;

import android.util.DisplayMetrics;

import android.util.Log;

import android.view.Window;

import android.view.WindowManager;

import android.widget.Toast;

import software.idis.rc\_porsche.MainActivity;

public class ControlActivity extends Activity {

volatile boolean shutdown = false;

private Thread bluetoothThread;

class BlueToothThread extends Thread

{

public void run() {

while(true) {

try {

int value = 0;

value = ControlsView.currentSendvalue.get();

Log.wtf("hello", "value is "+ value);

OutputStream outStream = MainActivity.clientSocket.getOutputStream();

outStream.write(value);

Log.wtf("hello", "sended");

if(shutdown == true || MainActivity.masterExit == true)

{

outStream.close();

break;}

Thread.sleep(10);

}

catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// for permanentrly landscape orientation

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);

// for fullscreen view

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN);

// disabling title

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

DisplayMetrics metrics = new DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(metrics);

ControlsView.displayMetrics = metrics;

ControlsView view = new ControlsView(this);

setContentView(view);

// enable bluetooth

bluetoothThread = new BlueToothThread();

bluetoothThread.start();

}

@Override

protected void onDestroy() {

// TODO Auto-generated method stub

super.onDestroy();

shutdown = true;

try {

bluetoothThread.join();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

// @Override

// public void onBackPressed() {

// // TODO Auto-generated method stub

// super.onBackPressed();

// shutdown = true;

// try {

// bluetoothThread.join();

// } catch (InterruptedException e) {

// // TODO Auto-generated catch block

// e.printStackTrace();

// }

// this.finish();

// }

}

**Class TrianglePath.java**

package software.idis.rc\_porsche;

import android.graphics.Path;

import android.graphics.Point;

public class TrianglePath {

Point a, b, c;

Path path;

public TrianglePath(Point one, Point two, Point three) {

a = one;

b = two;

c = three;

path = getNewPath(a, b, c);

}

private Path getNewPath (Point a, Point b, Point c)

{

Path temp = new Path();

temp.setFillType(Path.FillType.EVEN\_ODD);

temp.moveTo(a.x, a.y);

temp.lineTo(b.x, b.y);

temp.lineTo(c.x, c.y);

temp.lineTo(a.x, a.y);

temp.close();

return temp;

}

public Path getPath()

{

return path;

}

}

**Додаток В**

**Система в сборі**



