Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Software Development for Android Devices

ENCS 539

Project #6

Walk Planner

Student Name: Mohammad Kabajah

ID: 1110600

Instructor's Name: Stephen Taylor

Section #:1

Date: 23/12/2014

The features of the application

You will be able to move the map in any direction, east west, north south by touch motions, you will be able to zoom in and zoom out that will let you magnify the image or make it smaller. You can see that at the bottom of the map that the data provider is mentioned. There is a text view that show the MAC address of the router you are connected to and another text view show the location of that router so for example if you are connected to the router in the IT building it the location text view will be showing IT Building which indicating where are you connected. If you are trying to go beyond the map of the campus you can't, it don't allow you to go beyond the borders of the map. There is a refresh button that is used to check the MAC address of the router that you are connected to and give you the MAC address and the location of that router, it didn't work once you lunched the app you need to click the button since you can't call the on click method in API level less than 15, and I didn't make it refresh automatically with constant period of time, because when I tried it with periodically checking it consumed so much of the battery it took around 5% from the battery in around 5 minutes so I consider make it manually refreshing so it will be less power consumption.

Here is the source code followed by some screen shots of the application while running and all the files will be added in the archive.

```
package edu.birzeit.fall2014.encs539.id1110600.walkplanner;

/**
 * This class is used to create an object for each router containing the BSSID
 * of the router and the location of that router
 * **/
public class Node {
	private String BSSID = "00:00:00:00:00:00";
	private String location;

	public Node(String BSSID, String location) {
		this.BSSID = BSSID;
		this.location = location;
	}

	String getBSSID() {
		return BSSID;
	}

	String getlocation() {
		return location;
	}
}
```

```
package edu.birzeit.fall2014.encs539.id1110600.walkplanner;

/**
 * This class is made to hold the BSSID of the router and the location of the
 * router each router node contain the router unique BSSID and the location of
 * the router Unfortunately i could not get all the BSSID's of all the routers
 * in the campus and all the buildings all the BSSIDs here is from the IT
 * Building but the last one is the one in my home i added it just for testing
 * purposes
 *
```

```
 * it would be better if this data was read from a file and stored in a hash
 * table or stored in SQL lite database it would be more efficient and better
 * and give the option to add new routers later or modify later but
 * unfortunately i didn't do it because i didn't have enough time
 * **/
public class Nodes {
	final static Node IT1 = new Node("FC:0A:81:10:A6:82", "IT Bilding");
	final static Node IT2 = new Node("FC:0A:81:52:76:B2", "IT Bilding");
	final static Node IT3 = new Node("FC:0A:81:52:6C:D2", "IT Bilding");
	final static Node IT4 = new Node("FC:0A:81:10:A6:80", "IT Bilding");
	final static Node IT5 = new Node("FC:0A:81:52:76:B0", "IT Bilding");
	final static Node home = new Node("80:1F:02:93:BD:E6", "Home");

	final static Node[] nodes = new Node[6];

	/**
	 * initialise the array that hold all the routers and their locations
	 * **/
	static void init() {
		nodes[0] = IT1;
		nodes[1] = IT2;
		nodes[2] = IT3;
		nodes[3] = IT4;
		nodes[4] = IT5;
		nodes[5] = home;

	}

	/**
	 * this method is used to give the location of a specific router based on
	 * the BSSID of it and return the location of that router
	 * **/
	static String getLocation(String BSSID) {
		init();
		for (int i = 0; i < nodes.length; i++) {
			if (BSSID != null &&
BSSID.equalsIgnoreCase(nodes[i].getBSSID())) {
				return nodes[i].getlocation();
			}
		}

		return null;

	}

}


package edu.birzeit.fall2014.encs539.id1110600.walkplanner;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;
```

```
/**
 * This class is used to draw the map on the screen zoom in and out from the map
 * **/

public class MapView extends View {

	Bitmap bit = null;// The object that will hold the map
	Paint paint = new Paint();
	Paint paint2 = new Paint();
	Rect source = null; // this will be the portion of the map that will be
					// drawn
	Rect rect = new Rect();
	Rect destination = null; // this will be the object where to draw the map
							// and will
	// be always the same
	int realX = 0; // the real dimension of the map
	int realY = 0; // the real dimensions if the map
	int maximumWidth = 0;
	int maximumHeight = 0;
	int cutWidth = 0; // the dimensions of the portion that will be displayed
on
					// the screen from the map
	int cutHeight = 0; // the dimensions of the portion that will be displayed
					// on the screen from the map

	public MapView(Context context) {
		super(context);
	}

	public MapView(Context context, AttributeSet as) {
		super(context, as);
	}

	public MapView(Context context, AttributeSet as, int t) {
		super(context, as, t);
	}

	/**
	 * OnDraw method is used to print the map on the screen it first check if
	 * the bitmap file is set and not null while it is loading the application
	 * the bitmap file will be set if the the view is ready and if it is not
	 * ready then its dimensions will be 0,0 and then if we try to invalidate
it
	 * will not draw anything we keep waiting until it will be set
	 *
	 * this happened when the application is launched at first the map view is
	 * not ready yet so the dimensions is 0,0 so we have to wait to the map
view
	 * to take its size so then we can draw in it
	 * **/
	@Override
	public void onDraw(Canvas canvas) {
		super.onDraw(canvas);
		if (bit != null) {
			canvas.drawBitmap(bit, source, destination, paint);
			canvas.drawLine(1, 1, 1, getHeight() - 1, paint);
			canvas.drawLine(1, 1, getWidth() - 1, 1, paint);
			canvas.drawLine(1, getHeight() - 1, getWidth() - 1,
					getHeight() - 1, paint);
```

```
                    canvas.drawLine(getWidth() - 1, 1, getWidth() - 1, getHeight()
- 1,
                                    paint);
                int h = getHeight();
              int w = getWidth();
             int size= R.dimen.textSize;

             String text = "Data provided by OpenStreetMap.org";
             paint.setTextSize(getResources().getDimensionPixelSize(size));
             paint.getTextBounds(text, 0, text.length(), rect);
             paint.setColor(Color.BLACK);
             canvas.drawText("Data provided by OpenStreetMap.org", w-
rect.width()-getResources().getDimensionPixelSize(size) , h-
getResources().getDimensionPixelSize(size), paint);
             }
      }

      /**
       * adjust the location of the map according to the moving gestures or the
       * swipe that the user do to view the map
       * **/
      public void adjust(int deltaX, int deltaY) {
             if (realX >= 0 && realX <= maximumWidth)
                    realX += deltaX;
             if (realX < 0)
                    realX = 0;
             if (realX > maximumWidth)
                    realX = maximumWidth;

             if (realY >= 0 && realY <= maximumHeight)
                    realY += deltaY;
             if (realY > maximumHeight)
                    realY = maximumHeight;
             if (realY < 0)
                    realY = 0;
             reDraw();

      }

      /**
       * this method is used to check on the parameters before drawing the map
and
       * if some of them is out of the boundaries to correct them to stay in the
       * boarders of the map dimensions and then invalidate the view
       * **/
      public void reDraw() {
             if (realX < 0)
                    realX = 0;
             if (realY < 0)
                    realY = 0;
             source.left = realX;
             source.top = realY;
             source.right = realX + cutWidth;
             source.bottom = realY + cutHeight;
             invalidate();
      }

      /**
       * this method is used to set the bitmap file of the map
```

```
         * **/
        public void setMap(Bitmap bitmap) {
                bit = bitmap;
                paint.setStrokeWidth(2);
                cutWidth = bit.getWidth() / 6; // used to get the width of the
portion
                                                                        // we want to
draw on screen with the
                                                                        // ration 1.5
which is the same ration
                                                                        // as the
original file
                cutHeight = bit.getHeight() / 4; // used to get the height of the
                                                                                // portion
we want to draw on screen
                                                                                // with
the ration 1.5 which is the
                                                                                // same
ration as the original file
        }

        /**
         * this method is used to set the portion of the map that we want to draw
on
         * the screen and the destination is always the same which is the size of
         * the map view
         * **/
        public void setRect() {
                destination = new Rect(0, 0, getWidth(), getHeight());
                source = new Rect(0, 0, cutWidth, cutHeight);
                maximumWidth = bit.getWidth() - cutWidth;
                maximumHeight = bit.getHeight() - cutHeight;
        }

        /**
         * this method is used to zoom in the map by decreasing the dimensions of
         * the source portion of the map and keeping the destination is the same
         * size as the view so it will give the effect of zoom in
         * **/
        public void zoomIn() {
                int oldX = realX + cutWidth / 2;
                int oldY = realY + cutHeight / 2;
                cutWidth = cutWidth / 2;
                cutHeight = cutHeight / 2;
                int newX = realX + cutWidth / 2;
                if (cutWidth < bit.getWidth() / 12) {
                        cutWidth = bit.getWidth() / 12;
                        cutHeight = bit.getHeight() / 8;
                        newX = cutWidth / 2;
                }

                if (newX != oldX) {
                        realX = oldX - cutWidth / 2;
                        realY = oldY - cutHeight / 2;
                }

                maximumWidth = bit.getWidth() - cutWidth;
                maximumHeight = bit.getHeight() - cutHeight;
                reDraw();
```

```
	}


	/**
	 * this method is used to zoom out the map by increasing the dimensions of
	 * the source portion of the map and keeping the destination is the same
	 * size as the view so it will give the effect of zoom out
	 *
	 * **/
	public void zoomOut() {
		int oldX = realX + cutWidth / 2;
		int oldY = realY + cutHeight / 2;

		int oldcutWidth = cutWidth;
		int oldcutHeight = cutHeight;

		cutWidth = (int) (cutWidth * 1.5);
		cutHeight = (int) (cutHeight * 1.5);
		int newX = realX + cutWidth / 2;

		if (cutWidth > bit.getWidth() / 1.5) {
			cutWidth = oldcutWidth;
			cutHeight = oldcutHeight;
			newX = cutWidth / 2;
		}

		if (newX != oldX) {
			realX = oldX - cutWidth / 2;
			realY = oldY - cutHeight / 2;
		}

		maximumWidth = bit.getWidth() - cutWidth;
		maximumHeight = bit.getHeight() - cutHeight;
		reDraw();
	}

}
```

```
package edu.birzeit.fall2014.encs539.id1110600.walkplanner;

import java.io.IOException;
import java.io.InputStream;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {
```

```
	Button refresh; // the refresh button
	TextView MACtv, Locationtv; // the textview of the MAC and location here i
                                                    // might say MAC but i mean
the unique BSSID
	WifiManager wm;
	MapView mapView;
	String noBSSID = "00:00:00:00:00:00";
	private float downX, downY, upX, upY;
	Button zoomIn;
	Button zoomOut;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		refresh = (Button) findViewById(R.id.refresh);
		zoomIn = (Button) findViewById(R.id.ZoomIN);
		zoomOut = (Button) findViewById(R.id.ZoomOUT);
		MACtv = (TextView) findViewById(R.id.MACtextView);
		Locationtv = (TextView) findViewById(R.id.LOCATIONtextView);
		mapView = (MapView) findViewById(R.id.board);

		refresh.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {
				/**
				 * this button will check the wifi network you are
connected to
				 * and get the BSSID of the router and print it on the
screen
				 * and then get the location of that router by its MACc
address
				 * and print it on the screen
				 * **/
				wm = (WifiManager) getSystemService(WIFI_SERVICE);
				WifiInfo temp = wm.getConnectionInfo();

				if (temp.getBSSID() == null ||
noBSSID.equals(temp.getBSSID())) {
					MACtv.setText("No Wifi connection");
				} else {
					MACtv.setText(temp.getBSSID().toLowerCase());
				}
				String loc = Nodes.getLocation(temp.getBSSID());
				if (loc != null) {
					Locationtv.setText(loc);
				} else {
					Locationtv.setText("UNKNOWN Location");
				}
				runOnUiThread(new Runnable() {
					@Override
					public void run() {
						MACtv.invalidate();
						Locationtv.invalidate();
					}
				});
			}
		});
```

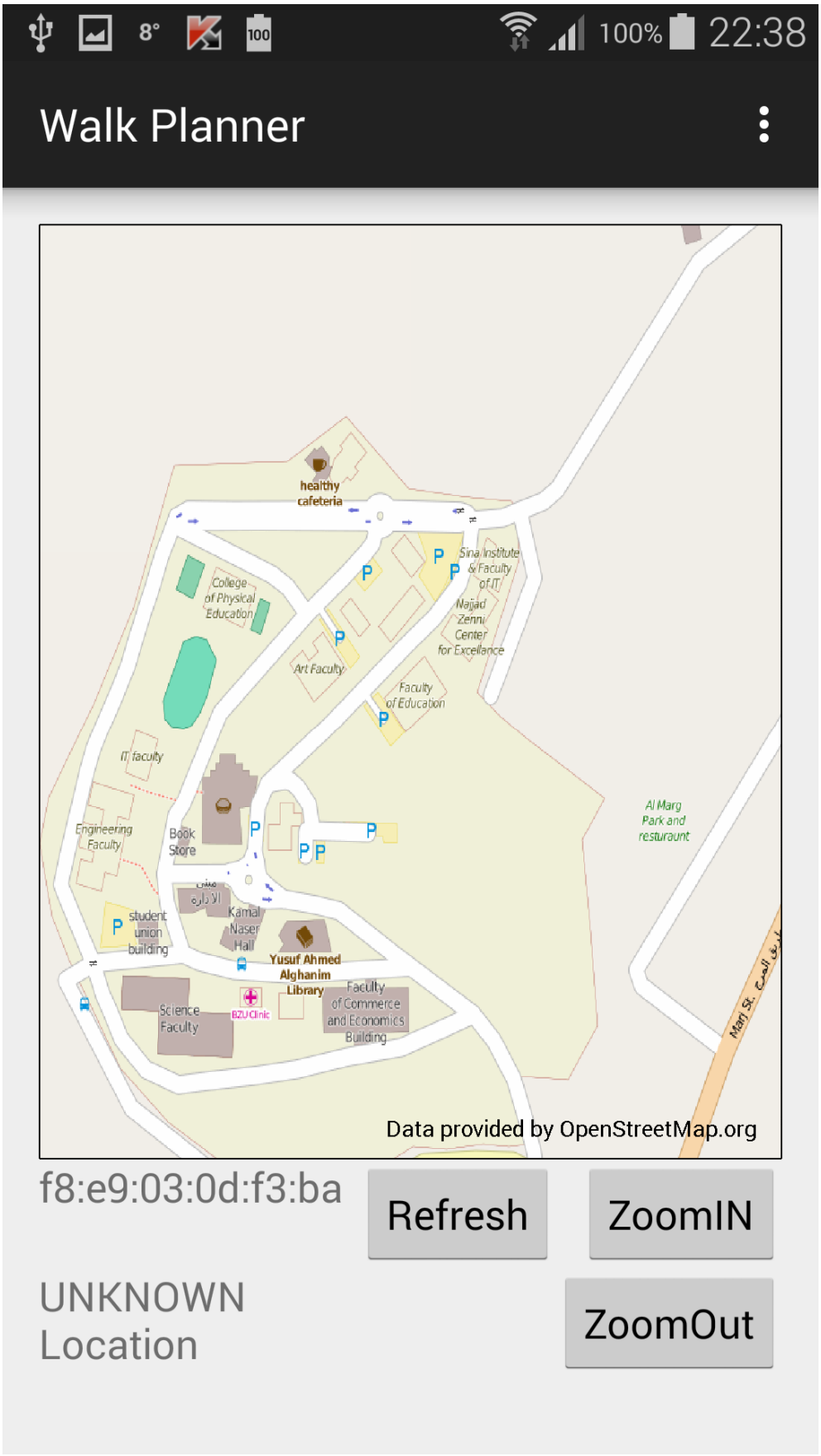
```
		zoomIn.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View view) {
				/**
				 * call the zoom in method in the map view class to do
the
				 * effect of zooming in
				 * **/
				mapView.zoomIn();
			}
		});

		zoomOut.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View view) {
				/**
				 * call the zoom out method in the map view class to do
the
				 * effect of zooming out
				 * **/
				mapView.zoomOut();
			}
		});

		/**
		 * Creating this anonymous thread to do simple job the it will end
this
		 * thread keep checking the dimensions of the map view if it is not
		 * created correctly yet it keep waiting until the map view is ready
		 * then it will try to open the map image from the Assets directory,
		 * decode the image to bitmap to send it to the map view to set the
		 * bitmap image that we will use to draw on the screen and to set
the
		 * source and destination rect objects that will be used to draw the
map
		 * after this thread complete this job and the map view is now ready
and
		 * the map is drawn on it the thread will stop/quit/terminated
		 * **/
		new Thread() {
			@Override
			public void run() {
				while (mapView.getWidth() == 0 || mapView.getHeight()
== 0)
					try {
						sleep(100);
					} catch (InterruptedException e) {
						e.printStackTrace();
					}
				try {
					InputStream bitmap = getAssets().open("map.png");
					Bitmap bit = BitmapFactory.decodeStream(bitmap);
					mapView.setMap(bit);
					mapView.setRect();
					// refresh.callOnClick(); //cannot be used in API
level less
					// than 15
```

```
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                        runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                        mapView.invalidate();
                                }
                        });
                }
        }.start();

        mapView.setOnTouchListener(new OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        /**
                         * here we are doing the swipe gestures, we get the
location of
                         * where the swipe start and the location of where the
swipe
                         * ended and calculate the difference between the X and
Y
                         * coordinates and send it to the adjust method to
adjust the
                         * map in the map view, in other words to change the
portion of
                         * what we see from the map
                         * **/
                        switch (event.getAction()) {
                        case MotionEvent.ACTION_DOWN: {
                                downX = event.getX();
                                downY = event.getY();
                                return true;
                        }
                        case MotionEvent.ACTION_UP: {
                                upX = event.getX();
                                upY = event.getY();

                                float deltaX = downX - upX;
                                float deltaY = downY - upY;

                                mapView.adjust((int) deltaX, (int) deltaY);
                                return true;
                        }
                        }
                        return false;
                }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
```

```
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			return true;
		}
		return super.onOptionsItemSelected(item);
	}
}
```

In the following screenshot we are connected to a known Wi-Fi network with known location

In the following screen shot we are connected to an unknown Wi-Fi network without known location



In the following screen shot we are not connected to any Wi-Fi network