Dogigator

Mitul Mahendrakumar Panchal, Shashank Kumar, Vrushti Shah, Yiyang Wang
{mpancha, sbkumar, vkshah, ywang95} @ncsu.edu

Abstract

Navigating from one place to another is a daily routine for many of us, however to do this without a sense of vision is a challenge many visually impaired people face. Our project is a navigation aid visioned to provide sufficient intelligence to Guide Dogs to help the blind person. Our application consists of two parts: an android application and a beaglebone black single board computer paired using bluetooth. The application can be installed on any android device with bluetooth and GPS capability. The beaglebone Black is placed on Dog harness which receives real time turn by turn navigation information and instructs Dog. Experimental Results show the effectiveness of the proposed system for navigation.

Introduction

We all take our eyesight for granted, but blindness is a disability that can bring life to a standstill as it leads to dependency on other individuals or objects. Even after years of technological innovation, sticks and guide dogs are one of the most widely used aids for blind people. The Dogs, even though trained, are very limited in their ability to help as they cannot lead a person to destinations unknown to him and this calls for widespread need of research and development in navigational systems for the blind. The researchers have shown that Dogs can follow instructions from embedded computer **[**1]. Our solution integrates the android mobile device with the Dog harness which instructs dog to move towards destination and hence develop a standalone navigation system for the visually impaired with help of guide dog. Hence, keeping this in mind we have created an Android application that could be used by a person to guide the guide dogs. It is a map based application that take the source and the destination inputs from the user, starts the GPS navigation and transmits the directions wirelessly to a beaglebone black attached to the dog's harness.

Related Work

Research is being done throughout the world to offer navigation system the best in terms of accuracy, safety and cost effectiveness. CyRAM, designed by K. Oto et al. [2], use to detect obstacles using ultrasonic sensors and inform the user using tension of a wire attached to the user. N. Bourbakis designed wearable navigation aid called Tyflos[3] comprising of wearable portable computer and sensors. There are other recently developed models called ROVI[4], PREDATOR, ,etc claiming to be more accurate. However with all these models, expense is a major drawback. Also since it required the user to continuously scan the environment, the communication between human and machine is time consuming.

To develop a navigation aid with both the above factors in mind i.e better response time and less expense, we decided to build upon work of Dr. Alper Buzkurt et al. who have developed a suite of communications to improve communication between dog and human beings. Guide Dogs are already being used by many visually challenged people to help them avoid obstacles, we decided to work on a project to use Guide Dogs to help navigate with real time instructions.

How we attack the problem

The navigation problem for blind person is solved by designing a mobile navigation system communicating turn by turn instructions to Guide Dog. As shown in a diagram a visually impaired person uses an android phone with Dogigator application installed and a Guide dog has a harness which is equipped with Beaglebone Black single board computer. The Dogigator application is an enhancement of open source map and navigation application OsmAnd[5]. The Google Map API could be used for generating navigation route but Google API requires the tracking and route recalculation algorithm to be developed by application developer. OsmAnd provides the complete navigation and tracking solution and it is open source. Hence we chose OsmAnd for developing our navigation application and developed a bluetooth communication module.

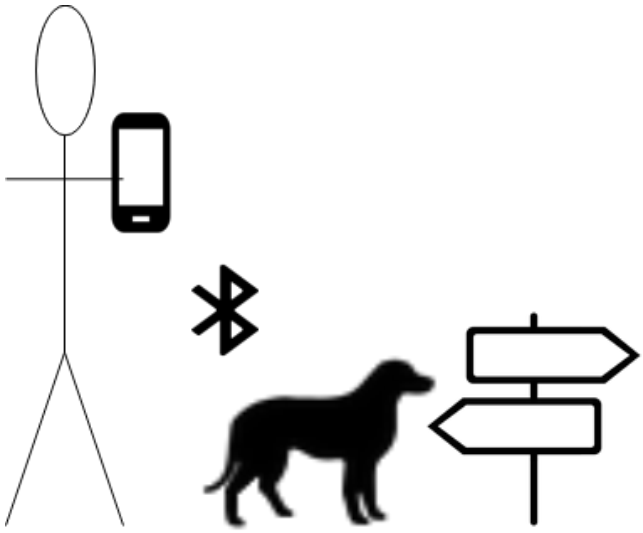


Figure 1: Guide Dog Navigation system

System Environment

The minimum requirement for the phone is that it should have minimum Android Kitkat 4.4 version and available disk space of 30 MB.
Our Dog Harness requires BeagleBone Black with Angstrom Operating system.
A USB Bluetooth dongle and Bluetooth services enabled on Angstrom.
A Pybluez python package to for bluetooth development module in python.

Design and Implementation

The high level design and development of the system is divided into following sub modules

1. Android Application Stub and Bluetooth Module
2. Dogi Bluetooth Receiver Module and turn Indication LEDs

1. Android Application Stub and Bluetooth Module

We developed a bluetooth connection module which opens a Bluetooth connection socket and provides an API to send asynchronous message to connected device. The bluetooth module provides an API to initialize the bluetooth connection during app startup. The module also provides an API to send an asynchronous message to connected

bluetooth device using specific UUID. The Routing module of OsmAnd is stubbed to send turn instructions to Beaglebone using send API provided by our bluetooth module. We adopted a client-server module for bluetooth communication and the mobile device acts as Client and sends connection request to the Bluetooth server running on Beaglebone. The following APIs describe the Bluetooth Module in detail.

BluetoothModule Class: The BluetoothModule class maintains the connection details in member variables and provides methods for bluetooth communication. This class provides following APIs

1. **startModule:** This function performs Bluetooth connection initialization. The following steps are performed by this API.

- *Get Bluetooth Adapter:* Obtains Device Bluetooth adapter object by calling the getDefaultAdapter API.
- *isEnabled:* Checks if Bluetooth is enabled on the device. If the bluetooth is not enabled it enables Bluetooth automatically. This required BLUETOOTH_ADMIN permission to be set in application manifest.
- *getBondedDevices:* Obtains the list of paired devices.
- Create Socket: Create a bluetooth socket using createRfcommSocketToServiceRecord function provided by android.
- *cancelDiscovery:* Cancel the ongoing discovery operation of bluetooth. This is very important step because without disabling the discovery bluetooth will spend lot of power
- *connect*: Connect to remote bluetooth server running on beaglebone.
- *Obtain Output stream:* Obtain output stream from connected bluetooth socket and save it in mmOutStream member variable.

2. send: Calls a write method to send message over connected socket. startModule API has to succeed before calling send.

3. cancel: API to close the socket connection.

VoiceRouter Stub: The VoiceRouter class updateStatus method provides the turn by turn information according to our code understanding. We created an object of BluetoothModule Class in constructor of VoiceRouting. Hence on application startup the BluetoothModule Object will be instantiated.

The Mobile device acts as Bluetooth client and sends the connection request during application startup. After the object instantiation, startModule API of the created object is called to open bluetooth client socket and connect to remote server.

...
protected BluetoothModule bluetoothMod = new BluetoothModule();
protected OutputStream bt_handle = bluetoothMod.startModule();
...

bt_send_to_doggi : This API translates the turn information from OsmAnd format to the simple string based message like left, right etc. After translating the turn message it sends the message across by calling send API of BluetoothModule object.

updateStatus: This API is OsmAnd API which provides the navigation information to Audio device. We have stubbed this API and called bt_send_to_doggi API to send the commands to beaglebone.

2. Dogi Bluetooth Receiver Module and turn Indication LEDs

The Dog Harness consist of beaglebone black single board computer. We are using USB bluetooth dongle to enable bluetooth radio on Beaglebone. We are using RFCOMM protocol for communication. The RFCOMM protocol emulates the serial cable line settings and is used for providing serial data transfer. RFCOMM runs on top of the Bluetooth's L2CAP layer and emulates a serial cable setting and status of an RS-232 port. The dogi_server.py is the python module which opens a bluetooth socket.

The beaglebone runs angstrom operating system. We have added a new service called "Dogi" which runs on beaglebone startup. Please refer the appendix section for steps to enable a new service in angstrom. The "Dogi" service runs the dogi_server.py python module. The dogi_server.py module is described below:

dogi_server Module:
We are using pybluez python package which provides bluetooth module. The bluetooth module is imported to use the bluetooth APIs. The pybluez module allows to create a bluetooth socket and bind to specified port. The dogi_server creates a socket and listens for connections. During initialization of Dogigator android application the mobile device send the connection request. The dogi_server accepts the connection and makes a socket receive blocking call. When the Dogigator application sends the turn instruction, the receive API unblocks and calls the turn function which blinks the beaglebone LED indicating the turn instruction for Dog.

Experimental Results

The system is tested with route simulation feature of OsmAnd application. The simulation plugin in OsmAnd allows the simulation of navigation without physical motion and hence allows us to test and debug our application. The debugging console ADB shell is used to view the log prints and status of our application. On Beaglebone, we open a terminal on SSH or USB serial interface and see the logs on beaglebone console. The following are the steps involved in testing the system.

- Build the application using gradle script and generate application apk.
- Install the application in Android device
- Extract the dogi_server module and run it on Beaglebone black.
- Pair the mobile device and beaglebone black via bluetooth wizard or hcitool.
- Start the application on Android device
- Monitor the LEDs on beaglebone and the console logs in adb shell and beaglebone terminal

The following screenshots show the experiment setup:

Figure 2: Dogigator Setup

The android mobile on the left in the image is running Dogigator android application which sends turn by turn commands to Bluetooth server on beaglebone back on the right. The image shows the right turn command being received by beaglebone. The circle shows the right most glowing LED indicating the right turn. In real dog harness, the vibrating motors convey the message to Dog. The debug prints on beaglebone console can be seen in screenshot below:

Figure 3: Beaglebone console screenshot

The following is the output of beaglebone angstrom active services showing dogi (dogi_server module) as active service

```
root@beaglebone:~# systemctl
UNIT                    LOAD   ACTIVE SUB              DESCRIPTION
dropbear@....7.1:52291.service loaded active running   SSH Per-Connection Server
gateone.service                loaded active running   GateOne daemon
dogi.service            loaded active running   Dogi
```

The following dump shows the logcat output on adb shell. The OsmAnd application dumps the real time log prints. We can see the BluetoothModule logs for connection setup.

```
root@mitulpanchal:~# adb shell
shell@m7:/ $
shell@m7:/ $ logcat -b main MITUL:V *:S
I/MITUL   (16585): BluetoothModule: startModule
I/MITUL   (16585): MainActivity:onStart:UUID generated
I/MITUL   (16585): BluetoothModule: startModule: getDefaultAdapter success
I/MITUL   (16585): MainActivity:onStart: BT turned ON
I/MITUL   (16585): 00:1E:0A:00:00:39
I/MITUL   (16585): Address matched
I/MITUL   (16585): MainActivity:OnStart: bluetoothClientSocket created
I/MITUL   (16585): BluetoothModule: startModule: cancelDiscovery done
I/MITUL   (16585): MainActivity:OnStart:bluetoothClientSocket connected!!
I/MITUL   (16585): MAinActivity:OnStart: All set
I/MITUL   (16585): MAinActivity:OnStart: returning mmOutStream
```

Issues Faced

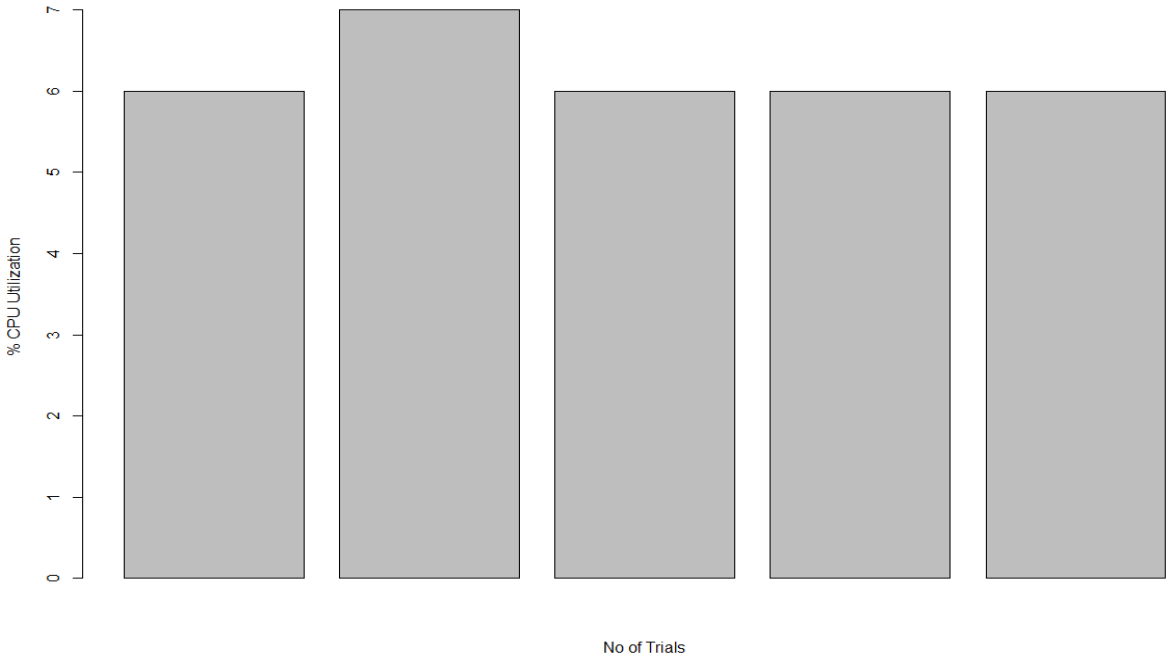
- Since OsmAnd is a very complex open source application, the stubbing of the application code and debugging was a challenge. Moreover the project is very large and hence requires gradle scripts to build. It does not work in Android Studio IDE which provides many debugging tools for android application. We solved the debugging issue by using the adb shell command (logcat). logPrinter class of android allows us to dump logs which can be viewed by logcat in real time.
- The USB power adapter which is provided with the beaglebone black does not provide sufficient power for driving USB Bluetooth dongle. We used the USB hub to provide the require current rating. For proper functioning of the attached USB device on beaglebone black, it requires the power rating of at least 2A.
- We tested the Bluetooth communication module in standalone application. The application was able to connect to bluetooth server running on beaglebone black but when we integrated the module in OsmAnd application the connection was failing. The reason for failure was the ongoing Bluetooth discovery operation performed by Bluetooth adapter. We found that it is necessary to stop the ongoing discovery operation by using cancelDiscovery() function before connecting to the bluetooth server. This is due to the fact that discovery operation is a very heavy operation. Also we require BLUETOOTH_ADMIN permission to be added in application manifest file to cancel the ongoing discovery operation.

Analysis

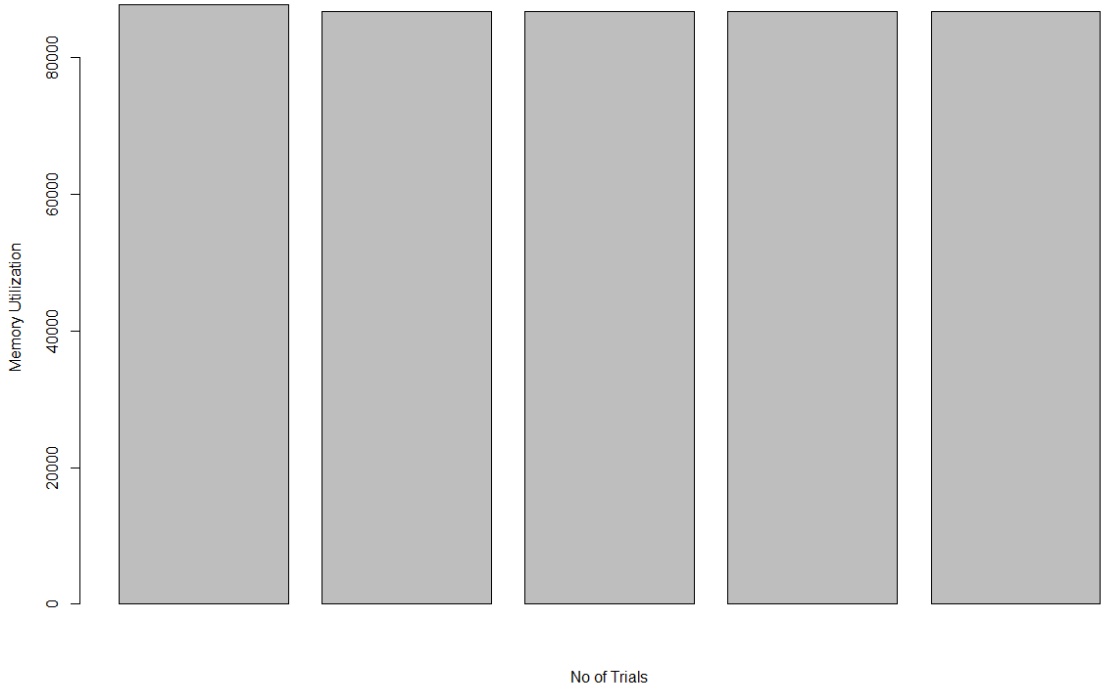
Our navigation aid works once the android phone and beagle bone black have been paired using Bluetooth socket and user enters his destination. The directions provided to beaglebone black is now being demonstrated using LEDs in absence of an actual harness. Once we have an actual harness, we would need to make sure that signal sent to beaglebone black would be enough for running the motors to indicate turns to dog.

After customising OsmAnd we wanted to be sure that the CPU and memory utilization of the Phone before and after starting the navigation application did not increase manifolds.
Below graphs are for OsmAnd application without starting navigation:

CPU Utilization vs No of Trials

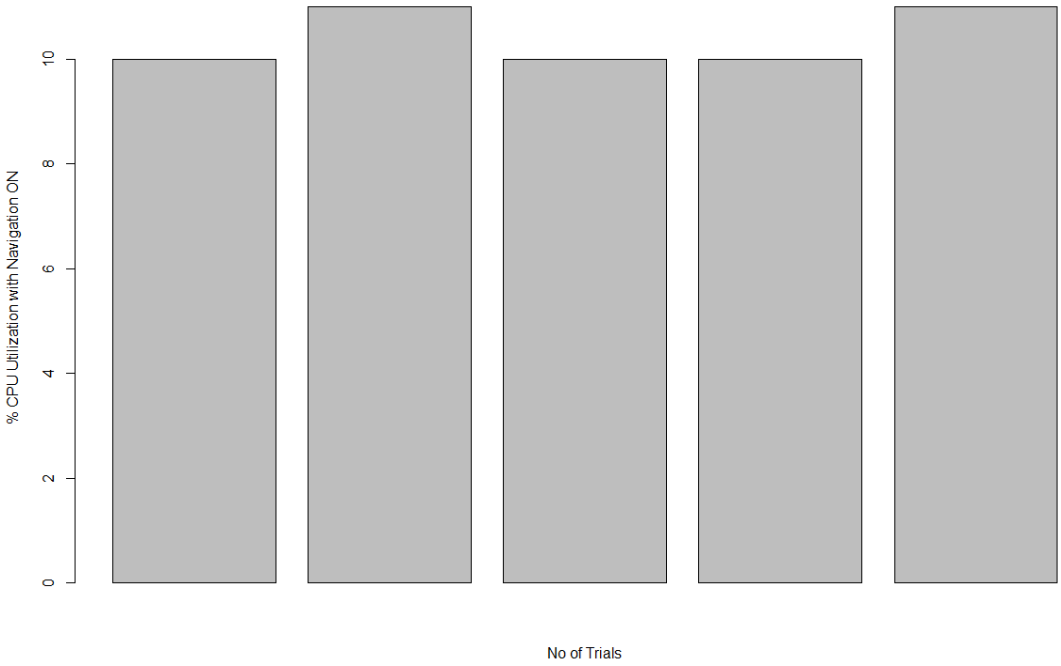


Memory Utilization vs No of Trials



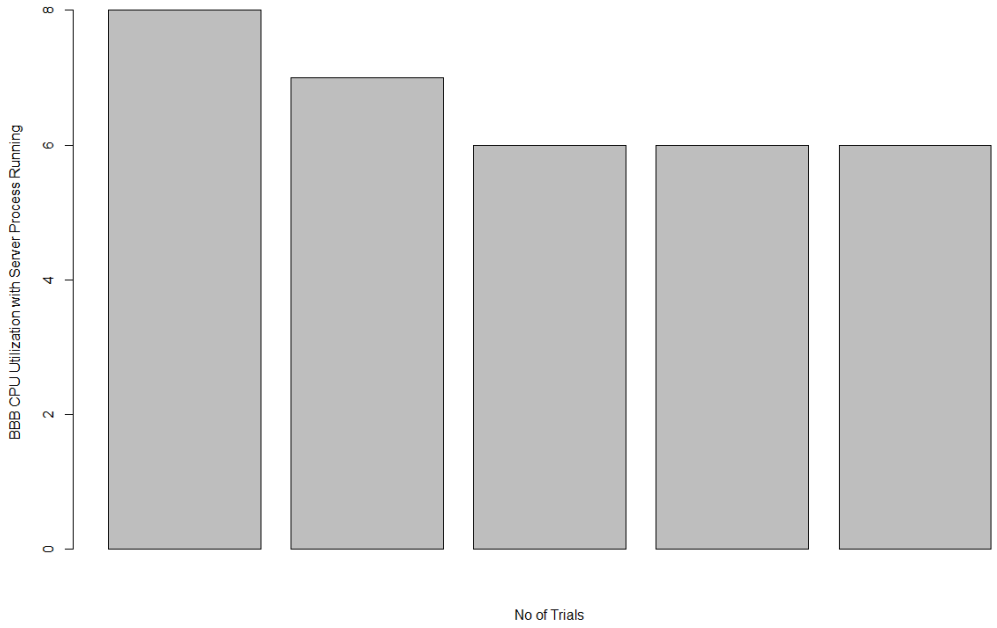
The below graphs are plotted for OsmAnd application with Navigation running and using our BluetoothModule:

CPU Utilization vs No of Trials



Memory Utilization vs No of Trials

We also measured the CPU utilization of beaglebone to make sure our server application did not over utilize the board CPU.



Comparing the end product to the problem description, we as a team were to successfully achieve what we set out for. We were able to successfully integrate our application with beaglebone black and send directions to the user chosen destination which is eventually meant to be translated to dog's harness. We were able to get a good response time in communication between the devices. Since our application is for visually challenged people, it would be absolutely necessary to add voice recognised inputs to enter the destination before navigation, which would be our future work. It would also be a good idea to have the application connected to an access point to monitor actual routes taken by blind person to increase his safety.

Conclusion

The day to day navigation could be challenging for visually impaired people. We designed a navigation aid for visually impaired person which is cost effective and does not require special training to blind person. We showed that the application successfully transmits step by step directions to the desired location to the dog harness wirelessly. The implementation of techniques to translate the signals from beaglebone to the dog are still in their experimental stage and further research needs to be done before the application can be used at the commercial level. In order to enhance the application in future we plan to improve the application by enabling voice recognition to enter the source and the destination. Moreover, the future holds scope to improve the ways to communicate different and complex type of directions to the dog.

Appendices

Enabling Bluetooth service on Beaglebone black

By default the bluetooth service is disabled on beaglebone black. The bluetooth service can be enabled by executing following commands on beaglebone console.

echo –e “\n[Bluetooth]\nEnable=true” >> /var/lib/connman/settings
systemctl enable bluetooth.service
systemctl start bluetooth.service

Enabling custom service in Angstrom
Create a new .service file in /lib/systemd/system/
eg: dogi.service
cat dogi.service
[Unit]
Description=Dogigator service

[Service]
WorkingDirectory=<script path>
ExecStart=/usr/bin/python dogi_server.py

[Install]
WantedBy=multi-user.target

ADB shell commands for obtaining debug console of android device
adb shell - This command connects the debugging shell to the connected android device
logcat -b main OsmAnd,MITUL:V *:S - This command prints the logs in real time only with the provided tags (OsmAnd and MITUL)

Reference
[1] Bozkurt, Alper, et al. "Towards Cyber-Enhanced Working Dogs for Search and Rescue." (2014).
[2] Ito, K., Okamoto, M., Akita, J., Ono, T., Gyobu, I., Tagaki, T., Hoshi, T., Mishima, Y.: CyARM: an alternative aid device for blind persons. In: Proc. CHI05, Portland, OR, 1483–1488 (2005).
[3] Bourbakis, N.: Sensing surrounding 3-D space for navigation of the blind. IEEE Eng. Med. Biol. Mag. 27(1), 49–55 (2008)
[4] Melvin, A., Prabu, B., Nagarajan, R., Illias, B.: ROVI: a robot for visually impaired for collision free navigation. In: Proceedings of the International Conference on Man-Machine Systems (ICoMMS), Batu Ferringhi, Penang, Malaysia (2009)
[5] OSM Automated Navigation Directions open source project. GitHub: https://github.com/osmandapp/Osmand
OsmAnd Home Page, Web: http://osmand.net/