# Implementation of a map route analysis robot: combining an Android smart device and differential-drive robotic platform

Chi-Hung Tseng[1], Ching-Chang Wong[1] and Wen-Hsuan Kuan[2]

[1]*Department of Electrical Engineering, TamKang University, 151, Yingzhuan Rd., New Taipei City, Taiwan*
[2]*Department of Applied Physics and Chemistry, University of Taipei, 1, Ai-Kuo W. Rd., Taipei, Taiwan*

**Abstract.** This paper proposes an easy-to-implement and relatively low-cost robotic platform with capability to realize image identification, object tracking, and Google Map route planning and navigation. Based on the JAVA and Bluetooth communication architectures, the system demonstrates the integration of Android smart devices and a differential-drive robotic platform.

## 1 Introduction

Robotic applications have attracted a great deal of attention in four areas: image identification and object tracking (IDOT) [1-2], map positioning and navigation (MPN) [3-4], web communication (WC) [5-6], and vocal control for speech recognition (VCR) [7-8]. In IDOT, feature-based or appearance-based recognition algorithms are usually adopted for facial recognition, road marking, and vehicle tracking. For better accuracy and reliance, the combination of autonomous self-deployment sensor networks with mobile robots [9] allows for the integration and incorporation of map route analysis, obstacle detection and avoidance. Parallel to the visual control of robots, vocal-sensitive robots designed to adaptively characterize the fundamental frequency and signal envelope of a sound [10] may greatly support surveillance and care services. Conducted by machine learning, recent advances in VCR for semantic analysis [11-12] and emotion analysis have achieved more realistic intelligence in human-computer dialogue.

These applications indicate that big data information [13] and WC technologies form the bedrock upon which innovative solutions of the Internet of Things [14] are being built, and are pushing forward the development and service of robotics. However, the aforementioned robotic applications also indicate that robots are able to be well placed in highly dynamic environments and can simultaneously feedback rich information. The cost of the huge computational efforts is, however, prohibitive, let alone extending them to outdoor or large-scale scenarios [15] where expensive beacon and differential-GPS [16] are commonly used at present. Therefore, the use of heterogeneous technologies to improve their performance, or at first at least to improve their affordability, has become an emergent and practical issue.

In practice, the strategy of installing the above four sensors or extended modules on robotic platforms is known for its lack of a price advantage. Meanwhile, there is too great a loading on the robot's on-board controller for it to support multidisciplinary control, especially for image processing. The realization of real-time data acquisition and analysis on a 16-bit processor would be tough. As a comparison, the state-of-the-art microelectronic technologies have been built to easily resolve 20 fps ultrafast and 640 ×480 2D real-time video streaming by the quad-core 64-bit processor with 1.5 Ghz clock speed of a smart phone. Nevertheless, the robotic microcontroller utilities remain significant in serving as middleware to bridge I/O pines, joints and communication protocols in the control of sensors and actuators via a smart phone. We expect that this short description has provided, either from an economic or performance perspective, a persuasive and broad motivation to investigate the combination of a mobile driving robotic platform such as the LEGO or Arduino suites with smart phones.

Due to the space limitation, in this report we focus specifically on introducing some studies on IDOT and MPN. We used the LEGO NXT robotic platform to build the main body of a mobile robot. A JAVA-based object-oriented algorithm was used to develop the programs on Android smart phones. The machine-human interaction was mediated by Bluetooth transmissions.

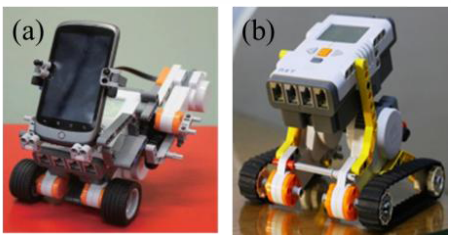


**Figure 1.** Two LEGO NXT mobile robotic architectures.

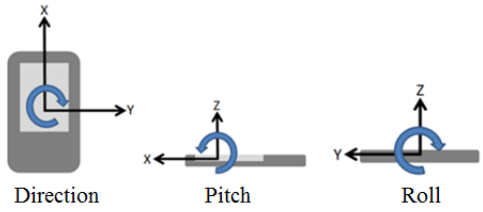


**Figure 2.** Orientation control of a smart phone.

## 2 The Architecture

While traditionally using differential two-wheeled robots encountered limitations of movement and direct control, these shortcomings were overcome by Ribeiro et al. who used a robot with three Omni-directional wheels [17]. The transverse drift induced by the axial-movement on that platform however leads to a reduction in the precision required in this work. Hence, a differential-driving mobile robot was adopted, considering the practical points of reducing the mechanical complicity and extensive accessibility. Translational displacement and the speed signals of the robot were provided via fast-response DC encoder servomotors. Two LEGO NXT mobile robotic architectures, as shown in Figure 1, were adopted in our tests. While Newtonian kinematics assumes that optimal rolling may occur without slipping, destructive pure rotations were found in some tests to irregularly induce a translational mismatch using the two-wheeled robot (a). Therefore, a tracked robot (b) with higher resistance contact was also implemented as a comparison. The environment errors were statistically reduced via multiple measurements and featured the one with better performance.

Due to its resource advantages, NXT Direct Commands provide Bluetooth protocols in direct communication with the robot [18]. The robot can interpret the commands embedded in the Direct Commands from the smart phone and then execute the corresponding action. This brings great convenience in robotic control to developers who can pay attention to programs for smart phones rather than for the robot. The communication platform is supported by leJOS NXJ, a JAVA virtual machine executed on NXT, for which many application objects can be freely accessed and developed on Eclipse [19]. Based on these specifications, a JAVA-based control kernel is appropriate for bridging the smart device of the Android core and the LEGO robot.

The operation of the robot was tested in two ways: (i) manual and (ii) autonomous. In case (i), the action control on the speed of the wheels was realized via an orientation sensor by detecting the variation in the orientation of the smart phone with pitch and roll (Figure 2), and via the displacement of the touch point (Figure 3). The control of the differential-drive is depicted in the right 4-quadrant circle, by assuming 4 definite movements for the left and right wheels as (v,v), (-v,-v), (v,-v), and (-v,v). In this manner, the speed of the individual wheels ($v_R$ or $v_L$) is definitely returned whenever the (x,y) coordinates of a specific (yellow) point are acquired. On the other hand, in case (ii), the robot is completely directed by the route planning accomplished with the importation of the Google Earth-supplied KML (Keyhole Markup Language) into the Google map [20].

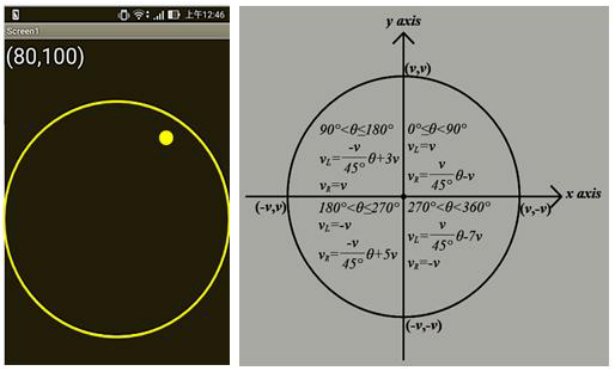


**Figure 3.** Touch control and 4-quadrant speed conversion.

```
import lejos.nxt.*;
import lejos.robotics.RegulatedMotor;
import lejos.robotics.subsumption.*;
import lejos.robotics.navigation.DifferentialPilot;
import lejos.robotics.navigation.RotateMoveController;
import lejos.util.PilotProps;
public class LineFollower {
public static void main (String[] aArg)
throws Exception
{
  PilotProps pp = new PilotProps();
  pp.loadPersistentValues();
  float wheelDiameter = Float.parseFloat(pp.getProperty(PilotProps.KEY_WHEELDIAMETER, "4.96"));
  float trackWidth = Float.parseFloat(pp.getProperty(PilotProps.KEY_TRACKWIDTH, "13.0"));
  RegulatedMotor leftMotor = PilotProps.getMotor(pp.getProperty(PilotProps.KEY_LEFTMOTOR, "B"));
  RegulatedMotor rightMotor = PilotProps.getMotor(pp.getProperty(PilotProps.KEY_RIGHTMOTOR, "C"));
  boolean reverse = Boolean.parseBoolean(pp.getProperty(PilotProps.KEY_REVERSE,"false"));
  Behavior DriveForward = new Behavior()
 {
    public boolean takeControl() {return light.readValue() <= 40;}
    public void suppress() {   pilot.stop(); }
    public void action() {
          pilot.forward();
          while(light.readValue() <= 40) Thread.yield(); //action complete when not on line   }   };
```

**Figure 4.** JAVA-based leJOS protocol for the LEGO NXT robotic platform, a partial demonstration of object tracking.

## 3 Image Identification & Object Tracking

In this section we demonstrate the results of the light tracking robot. A part of the JAVA-based leJOS program is shown in Figure 4, and three screenshots from a video are shown in Figure 5. IDOT on obstacle and marking detections was implemented via an on-board library, openCV [21], a very powerful open source for image identification. As the coordinates of the object are read, the smart device is able to estimate the relative position between the object and the robot. This allows the robot to adjust its movement. Besides, the captured image provides the information of the object size by which the relative distance between two subjects can also be estimated and conducted to determine the correspondent moving speed. The combination of image identification and dynamic fine tuning directly leads to successful improvement in obstacle avoidance in the Google map routing.

**Figure 5.** Image identification and light avoiding realized via openCV calculation.

In Figure 5, the robot was designed to avoid illumination from a flashlight. The idea was realized by openCV calculation, incorporating algorithms of image binarization and multiscale edge detection. The theoretical evaluation of the coordinates and the size of the light return the information of the direction and distance to the following movement of the robot.

In our experiment, it was found that the homogeneity of the illumination has a direct influence on the accuracy of the image identification. This problem may possibly be fixed by adopting a Kalman filter or particle filter algorithm to speed up the reaction of the smart phone to external light sources. Further verification of this reasoning and extended investigation will be reported elsewhere in the near future.

## 4 Map Positioning & Navigation

In this section we demonstrate the integration of Google map navigation and mobile robot control via the Android platform. As shown in Figure 6, the experiment was performed by first choosing the starting point and the destination on the Google map. The GPS returned latitude $B$ and longitude $\lambda$ of the objects in terms of [22]

$$B = \cos^{-1}\sqrt{\frac{b^2 - Z^2}{b^2 + Z^2 e'^2}} \qquad \lambda = \cos^{-1}\frac{X\sqrt{1-e^2\sin^2 B}}{a\cos B} \tag{1}$$

on the reference of the geodesic coordinates, in which *a(b)* is the semi-major(minor) axis of the Earth, XYZ are Cartesian coordinates, and $e = \sqrt{a^2 - b^2}/a$ and $e' = \sqrt{a^2 - b^2}/b$ are the first and second eccentricities of the spheroid. The arc length $d$ that the robot has travelled was evaluated via the Haversine formula [22]:

$$d = 2R \times \sin^{-1}\left(\sqrt{\sin^2\left(\frac{B_2 - B_1}{2}\right) + \cos B_1 \cos B_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{2}$$

regarding $R$ the radius of the right sphere Earth, $B_{12}$ and $\lambda_{12}$ represent the geographical latitude and longitudeof points 1 and 2. Then the corresponding displacement and rotation angle of the robot can be calculated by applying the commands bearingTo() and distanceTo() classified in the Location of Android APIs (application programming interface).

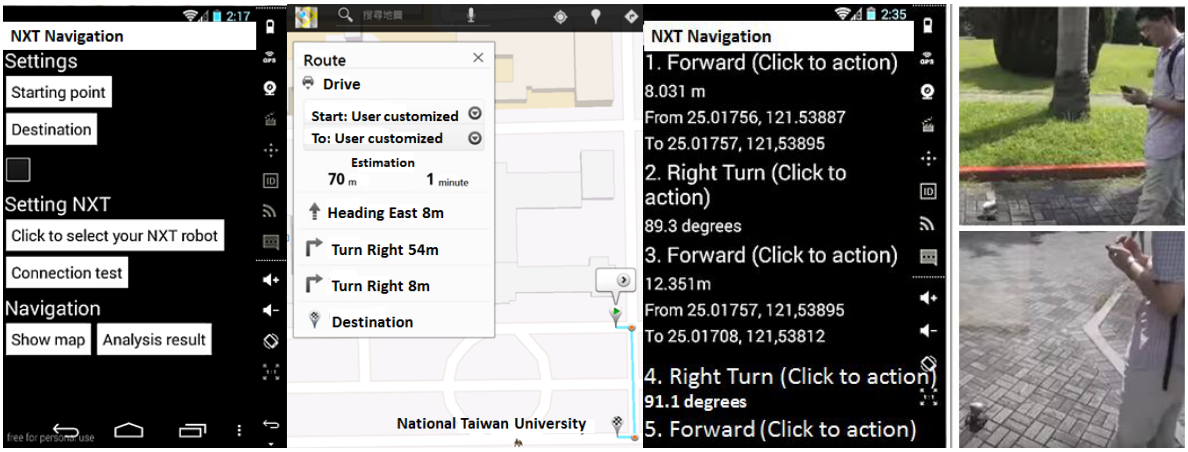


**Figure 6.** Google map route planning and navigation of a mobile robot. From left to right: Android-NXT communication interface set-up display, route planning information, and moving instruction.

To reach a precise measurement, an infinitesimal displacement within an extremely slow movement, although inconvenient, is established. For practical use, inevitable zig-zag overshooting and the cumulative error have to be considered. We applied a compass sensor with PID controller (proportional–integral–derivative) for stepwise error correction. Eventually an excellent refinement was revealed in the dynamic control.

Furthermore, we also investigated the data transmission between the smart phone and the robot. The

experiment was designed by transferring the detection readings of a supersonic sensor carried by the robot to the Android smart phone, and displaying the data in terms of various kinds of Google Charts. Figure 7 demonstrates a radar chart generated from a fixed point rotation, in which data acquisition was implemented every 45°. The path on which the chart can be accessed is given byhttps://chart.googleapis.com/chart?cht=rs&chs=500x500&chd=t:**77,66,15,0,31,48,100,77**&chco=FF0000,FF9900&chls=2.0,4.0,0.0%7C2.0,4.0,0.0&chxt=x&chxl=*0:%7C0%7C45%7C90%7C135%7C180%7C225%7C270%7C315*&chxr=0,0.0,360.0&chg=25.0,25.0,4.0,4.0&chm=B,FF000080,0,1.0,5.0%7CB,FF990080,1,1.0,5.0%27, in which the numbers in bold type are the sample data and those in italics denote the spacing.

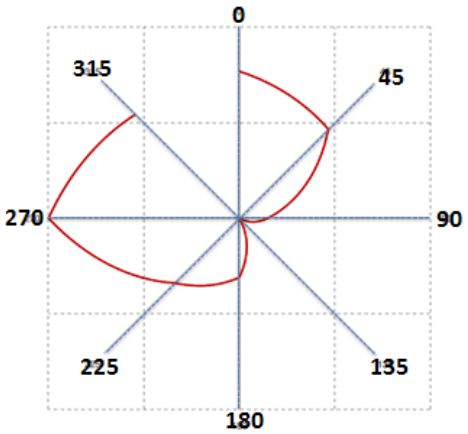


**Figure 7.** A radar chart. Environment data detected by supersonic sensor are shown by the red curve.

In our experiment, it was found that the web condition plays a primary role in the signal distortion and data reliability. Unfortunately, the signal transmission took roughly 0.3~0.4 seconds by Bluetooth communication, and another few seconds are required to the Cloud. Therefore, an optimal sampling and trigger has to be seriously considered. Moreover, the 45° sampling was found to reach the best recognition in our tests.

However, the impact of disturbance can be statistically reduced in long-time or multi-term measurements. Therefore, it is still worth noting that since real-time data storage is likely to soon be available on both the Cloud and smart phones, Google Charts will then provide fast and multivariate interfaces for big data processing and display. On the other hand, the combination of a smart phone reveals the power of synchronized remote control of the mobile robot and server end, bringing the potential of intensive applications to the industrial Internet of Things.

## 5 Conclusion

In this project we demonstrate image identification, object tracking, and Google Map route planning and navigation with an easy-to-implement and relatively low-cost robotic LEGO NXT platform. The robotic kinematics was under autonomous control based on the JAVA and Bluetooth communication architectures. The systematic connections on the leJOS, openCV and Android APIs applications lead to the realization of an integration of Android smart devices and a differential-drive robotics platform. Finally, we have demonstrated that environment detection by a mobile robot can be easily achieved. The combination of smart phone and Google Charts made it easy to realize big data storage and characteristic distribution. The use of smart phones for robotic control achieves a powerful integration of control, monitoring, communication, sensing, and calculation on a personal mobile device. Undoubtedly, strong connections between smart phones, the Cloud, and instruments must greatly inspire extensive and intensive creativity and imagination regarding innovative applications of cyber-physical integration.

## Acknowledgement

This work is partially supported by the Ministry of Science and Technology of Taiwan under MOST 104-2511-S-845 -009 -MY3. The authors would like to thank C. H. You, C. M. Yang and C. S. Chang for their help with the graphic editing.

## References

1. Ying Lan, Gangfeng Yan, and Zhiyun Lin, IEEE Transactions on Automatic Control, **56**, 1170 (2011).
2. F. Heidari and R. Fotouhi, J. Mechanisms Robotics, **7**, 041025 (2015)
3. Cheng Chen, Wennan Chai, and Hubert Roth, J Intell. Robot Syst., **80**, 365 (2015)
4. Haibin Yan, Marcelo H. Ang Jr., Aun Neow Poo, Int. J. Soc. Robot, **6**, 85 (2014)
5. Rafael Soca, Sebastián Dormido, Raquel Dormido and Ernesto Fabregas, Sensors **15**, 30076 (2015).
6. Jong-Hoon Kim, Gokarna Sharma, Noureddine Boudriga, S. Iyengar and Nagarajan Prabakar, EURASIP Journal on Wireless Communications and Networking 2015:262 (2015).
7. Naouel Ayari, Abdelghani Chibani, Yacine Amirat, and Eric Matson, Robotics and Autonomous Systems 75, 17 (2016).
8. J. Fernandes, J. Martinez-de-Dios, I. Maza, F. Fabresse, and A.Ollero, Int. J. Adv. Robot. Syst., 12, 70 (2015)
9. V. Kumar, D. Rus, S. Singh, IEEE Pervasive Comput. **3**, 24 (2004).
10. M. Kitani, T. Hara, H. Hanada, and H. Sawada, 3rd Conference on Human System Interactions, 728 (2013).
11. Miguel Oliveira, et. al., IEEE/RSJ International Conference on Intelligent Robots and Systems, 2216 (2014).
12. Li Zhang, Ming Jiang, Dewan Farid, and M.A. Hossain, Expert Systems with Applications **40**, 5160 (2013).
13. Ming Jiang and Li Zhang, Procedia Computer Science **53**, 141 (2015).

14. Luigi Atzori, Antonio Iera, Giacomo Morabito, Computer Networks **54**, 2787 (2010).
15. Christian Siagian, Chin-Kai Chang, and Laurent Itti, 2013 IEEE International Conference on Robotics and Automation, 564 (2013).
16. D. B. Wolf, C.L. Judy, E. J. Haukkala, and D. J. Godfrey, OCEANS 2000 MTS/IEEE Conference and Exhibition 1, 79 (2000).
17. F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, N. Pereira, Control 2004, University of bath, UK, September, 2004.
18. http://www.mindstorms.rwth-aachen.de/documents/downloads/doc/version-4.04/direct_commands.html
19. https://eclipse.org/
20. https://developers.google.com/kml/
21. R.W. Sinnott, Sky and Telescope, **68**, 159 (1984).
22. Hoffmann-Wellenhof, B. H. Lichtenegger, and J. Collins, *GPS: Theory and Practice*. 3rd ed. New York: Springer-Verlag (1994).