Modular Autonomous Robotics Platform for Educational Use

James LUMSDEN and Cesar ORTEGA-SANCHEZ
Electrical and Computer Engineering, Curtin University
Perth, Western Australia
c.ortega@curtin.edu.au

***Abstract*—Robotics is a field that continues to grow as robots become common in environments as varied as households and the battlefield. This paper presents a low cost robotics development platform using commercial off-the-shelf parts for educational and academic use. It is a direct response to the high cost and limited functionality of existing platforms. A navigation and obstacle-avoidance Fuzzy Controller is provided to accelerate the typical development process for a mobile robot. The fundamental aim is to facilitate future robotics projects by producing an inexpensive, modular and highly accessible platform that improves upon existing commercial offerings.**

Keywords- Educational robotics platform, autonomous navigation

I. INTRODUCTION

Robots have become more and more important in modern society as they can be found in environments as varied as households and the battlefield. This places an emphasis on research and development in the field of robotics. However, there is a serious lack of inexpensive yet highly flexible platforms for the development of mobile robots. This is a high barrier of entry to the robotics industry and makes it difficult to perform research without a large budget. Current platforms are not only too expensive but also limited in flexibility. Typically they are indoors-only and have little available physical space for expansion. It is also difficult to change hardware as they are not designed to be modified or expanded. This leads to a "reinvention of the wheel" whenever common functionality such as navigation or obstacle avoidance is required.

This paper proposes a low-cost, highly-flexible robotics platform for the development of small, autonomous ground vehicles. The platform uses commercial off-the-shelf parts that are readily available and inexpensive. The same principle was applied to software, with the focus on open source operating systems, libraries and development tools.

Section II provides an overview of modern robotics and discusses relevant background issues and existing platforms. Sections III and IV cover the hardware and software aspects of the platform, respectively. Section V presents the navigation and obstacle-avoidance fuzzy control system, while the verification and performance evaluation of the platform is presented in Section VI. Section VII contains conclusions and avenues for future work.

II. BACKGROUND

A. Development Platforms for Autonomous Robotics

Autonomous robots have to satisfy a number of, sometimes conflicting, requirements. Expected performance varies from one application to the next. To solve the problems of autonomous navigation and obstacle-avoidance not only the algorithms need to be understood, but also the premises behind their implementation. Consideration must be given to both the available resources and the goals of the robot itself. For most applications, a low-cost robot operating in a relatively simple environment will not gain any significant benefits from an overly complex control scheme. However, autonomous navigation and obstacle avoidance should be standard features additional to the robot's intended tasks.

Presently, prices for a single robot range from $400 for a LEGO™ Mindstorms NXT kit [1] up to $26 million for the latest unmanned aerial vehicle [2].

There are formal platforms to develop autonomous ground vehicles. For example, the Surveyor SRV 1 robot developed by Inertia Labs and Surveyor Corp which was designed for research, education and exploration [3]. However most of these platforms do not allow further expansion [4].

B. Control Systems

Conventional control systems are largely based on the development and analysis of mathematical models for physical systems [5], and then a closed-loop controller is designed using the model. Control is achieved using conventional approaches such as proportional-integral-derivative (PID), lead-lag and state feedback control [6]. The advantage of these systems is that they are well understood and widely used.

Non-conventional control systems can be classified into three types: Reactive, Deliberative and Hybrid. A reactive architecture is based on the implementation of actions commonly found in nature. A reactive system uses a direct mapping between stimuli and actions. These responses should co-operate to achieve or maintain a goal. However, these actions and conditions are already defined and cannot be considered a goal-based intelligent agent. In contrast, deliberative architectures involve constructing a plan and acting on it, while hybrid systems are a mixture of the two [8]. Examples of non-conventional, reactive control strategies are: Neural networks [9] and fuzzy systems [10].

III. HARDWARE FOR THE ROBOTICS PLATFORM

A. Selection Criteria

The selection criteria for the components used in a robot heavily depend on the requirements. The following specifications were drafted based on the desired attributes:

- The chassis must be at least 20cm x 20cm to provide enough room for multiple electronic boards.
- The robot must be capable of outside operation and is expected to operate on uneven terrain but does not need to be weather-proof.
- The steering system should be capable of turning in-place and is expected to use differential steering or similar.
- A standard 5V rail will be provided with at least 1A of available current for additional devices.
- A single set of batteries should provide power for all electronics and motors.
- A real-time operating system will be used to allow for a task-based modular design. It is expected that at least half of the processor time and code size should be available for additional development.
- Sensors are required for positioning and obstacle detection and should be capable of panning for further information.
- The microcontroller must support a number of communication protocols in order to maximize peripheral capability. This should include SPI, I2C, ADC and UART capability with hardware support preferred over software.
- Total parts cost for the robot should not exceed $1,000 USD

B. Chassis

Several companies produce a variety of chassis that could be used in the platform. After evaluating half a dozen, the chassis chosen was the Lynxmotion's A4WD1 [16]. It provides the largest amount of space and mounting options of all evaluated kits. It is built with aluminum brackets and laser cut panels to make it lightweight and easy to drill and modify, while providing high impact strength and weatherability [17]. In addition, the A4WD1 has the highest ground clearance which made it the most suitable for outdoor operation.

The A4WD1 is flexible enough to receive variety of motors and wheels. The current prototype uses four Lynxmotion's GHM-02 motors. Each motor can run up to 120 rpm with an integrated 50:1 reduction gear. Motors can provide a torque of 8.9 kg-cm, consuming 1.5 A @ 12V.

The selection of wheels available is fairly limited and primarily consists of rubber and foam varieties. While a number of sizes are produced the focus for this platform is on the larger variants. Coupled with the higher torque but slower GHM-02 motor, the larger wheels provide reasonable speed, torque and ground clearance. Figure 1 shows the A4WD1 with the selected wheels.

Figure 1. Lynxmotion's A4WD1chassis

C. Microcontroller (uC)

There are a multitude of microcontroller-based boards specifically designed for robotics applications available in the market - [17], [18] and [19]. The Pololu Orangutan X2 was selected for its performance, flexibility, expandability, and integrated peripherals. It uses Atmel's ATmega644 microcontroller @ 20 MHz and has 4 KB of SRAM and 64 KB of flash memory available [19]. This board was specifically designed for robotics; hence it provides a number of additional features such as an independent motor controller daughter board and a parallel LCD display connector. It also has 16 available user I/O pins, each accompanied by ground and power pins to simplify the connection of sensors and other devices and reduce the overall wiring complexity. The included VNH2SP30 motor driver is capable of delivering up to 14A continuous and 30A peak current. See Figure 2.

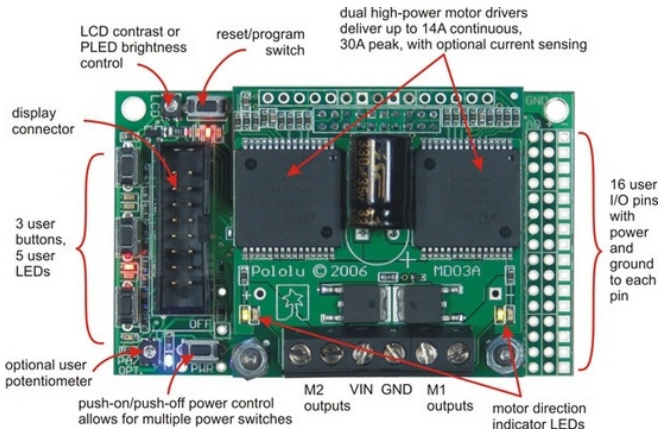

Figure 2. Top down view of the Pololu Orangutan X2 robot controller [19].

D. Distance and Obstacle-Detection Sensors

Sharp infrared rangers were selected to measure distance and detect obstacles. All Sharp infrared rangers utilize an infrared LED and a small linear charge-coupled device (CCD) array [20]. The CCD is able to determine the angle of incidence for a reflected beam and triangulate the distance to the object, making it particularly resilient to changes in ambient light

compared to other infrared sensors which rely on the amount of reflected light, rather than the angle. Figure 3 shows a photograph of the sharp IR rangers selected.

Figure 3. Sharp IR sensor variants and physical size [20].

E. Position Sensors

In this project the aim was for a low-cost, low-power device with minimal physical size. A GPS module with an update rate of 1Hz and a ceramic antenna was selected: The EM-406A GPS module from USGlobalSat [21]. It is a 20-channel GPS that uses the SiRFstarIII chipset for high sensitivity and indoor tracking [22]. Operating voltage is 5V with 44mA current draw. The entire module is 30x30x10mm in size and includes a ceramic antenna, which makes it ideal for maximizing the available physical space on the platform. At the time of writing, it is described as the smallest complete module available. It is shown in Figure 4 [23].

Figure 4. EM-406A GPS module from USGlobalSat [21]

F. Heading Sensors

A source of heading information that allows for an accurate heading to be obtained even while remaining stationary was required. An electronic compass was the logical choice.

Electronic compass modules utilize 2-axis magnetometers and a microprocessor to produce a heading based on the orientation of the earth's magnetic field [25]. The compass module selected was the Devantech CMPS03 module. It has been specifically designed for robotics applications and provides both an I2C and PWM interface [26]. It is one of the lowest cost modules available and runs at the same 5V as the other components in the system. Figure 5 shows the layout of the device and labels for pins.

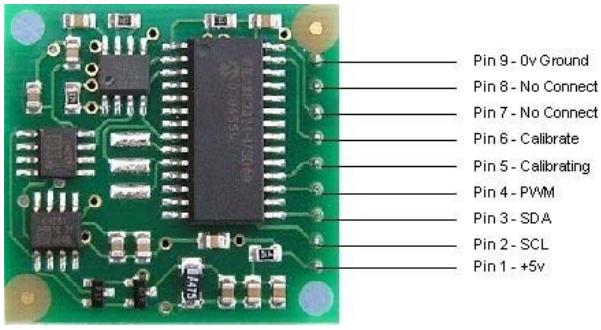

Figure 5. Devantech CMPS03 compass module [26]

G. Batteries

Several kinds of batteries are suitable for robotics applications. Each battery technology varies significantly in terms of energy density and cost. A comparison of relevant technologies can be found in [27], [28].

While NiCd batteries are inexpensive and have a relatively high cycle life, the memory effect is particularly severe. That is, when the battery is recharged after a partial discharge its capacity will be reduced. This also occurs with NiMH batteries but to a much lesser degree. Energy density of modern NiMH batteries can be as high as 80 Wh/Kg which makes them ideal for robotics. Hence, two 6V, 3000 mAh NiMH batteries connected in series were selected for the robotics platform.

Figure 6 shows a block diagram of the electronics in the robotics platform.

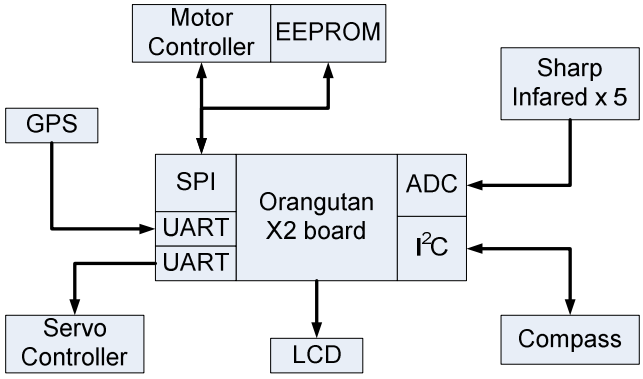

Figure 6. Block diagram of robotics development platform

In Figure 6, the SPI connection to the auxiliary microcontroller allows for the motors, EEPROM and a buzzer to be controlled without requiring any external connections. The Sharp IR sensors are connected to five of the eight analog to digital converter channels on PORTA. As the I2C interface is driven by software, the compass is able to use any of the available pins. The GPS and servo controller share a single UART. Data is received from the GPS on the RX pin, while the servomotors are moved by sending commands out of the TX pin. This enables both devices to be used with only a single serial connection and cuts down on cabling complexity.

IV. Software for the Robotics Platform

A. Development Tools

The software for the project was developed using AVR Studio, an Integrated Development Environment (IDE) provided by Atmel, the manufacturers of the AVR series of microcontrollers. AVR Studio is a free windows application described as a "project management tool, source file editor and chip simulator" [31]. However, AVR Studio does not include a compiler. An open source suite of development tools known as WinAVR is available and integrates with the IDE. This includes a GNU GCC compiler for C and C++ along with a programmer and debugger [32]. In addition, all of the internal registers and I/O devices can be viewed and modified for debugging and learning purposes.

B. Operating System

To simplify the implementation of control algorithms, FreeRTOS was mounted on the hardware platform. FreeRTOS is a portable, open source, royalty free, mini real time kernel [33]. More importantly, there are 23 official architecture ports and the operating system is well documented and supported with a free online reference manual and a number of inexpensive books available for purchase. One the available ports for the AVR ATMega323 was modified to run in the robotics platform. The only changes required for the ATMega644 involved modifying the timer setup for the kernel tick. This was demonstrated by Dalheimer using a slightly different platform [34]. In terms of functionality, the kernel provides pre-emptive, cooperative and hybrid scheduling options, as well as full support for a number of synchronization primitives such as semaphores. In FreeRTOS any task can delay its execution by fixed or dynamic intervals which allows for periodic execution of code with minimal effort [33].

The strength of real-time operating systems lies in their ability to split complex software into independent parts. FreeRTOS enables software to be divided into tasks based on responsibilities, frequency and priority. In the robotics platform, these tasks will be monitoring the various sensors, processing data and controlling actuators. As each task is scheduled separately and different sensors require processing at different times it is logical to create a task for each sensor type. This results in tasks which are naturally independent of one another and loosely coupled. Loose coupling is considered to be an indicator of good software design [35].

The distance sensors have an update rate of 20 Hz, therefore a task running every 50 ms was selected. The GPS update rate is only 1 Hz and it is constantly outputting data that must be either processed or buffered. A 1 Hz task would require a large buffer in order to store all the data for that period. In order to minimize the buffer requirements a frequency of 10 Hz was selected. The same frequency is also selected for the compass due to its 10 Hz internal update rate. Finally, the LCD is given a one second period as it is difficult to read quickly changing data and it does not perform a critical role in the system. As such it is also assigned the lowest priority. The highest priority is given to the distance sensors because obstacle avoidance is a strong requirement. Ordering of the GPS and compass tasks is not particularly critical; however as the GPS deals with streams and data buffers it should take priority because if a character is missed or the buffer overflows, then the GPS data will be lost and not repeated for another second.

V. Fuzzy Control System

This section describes the implementation of a fuzzy logic control system to solve navigation and obstacle-avoidance in the development platform. The control system combines data from both the navigation and obstacle avoidance subsystems in order to produce a steering direction for the motors. While it is not particularly difficult to create individual control systems for each subsystem, the problem lies with combining the results in a meaningful way. A simple method is to only enable navigation when no nearby obstacles are detected but this is a rudimentary approach. If a central obstacle is detected the movement vector should be biased towards the destination heading. This is not possible for a control system which switches between navigation and obstacle avoidance modes. What is required is a system that allows for various rules to be defined which govern behavior depending on infrared sensor distances and the heading offset. Fuzzy logic is well suited to this application as it allows for the rules to be defined using if-then constructs and linguistic variables. A good introduction to Fuzzy Systems can be found in [6].

In the design of the fuzzy controller membership functions were created for sensor angles, sensor distance, heading offset and steering angle. Rules were then constructed based on a common-sense approach. Closer objects result in a greater change to steering angle as an immediate response is required to prevent collision. Objects that are in front also require a greater response than those to the side. Steering will also adjust to the heading offset if that direction is free from obstacles. Table 1 shows the rules used to achieve this behavior.

The MATLAB fuzzy logic toolbox was used to test the membership functions and rule sets before they were used on the physical platform. This allowed for the response to be visualized and also enabled testing of different inference methods. Three-dimensional graphs were used to quickly analyze changes; however it is important to recognize the system is actually seven dimensional as it has six inputs and one output, and these graphs did not show the interaction of all variables. The interaction between all the rules is governed by the fuzzy logic inference method. The typical approach is to use max-min inference as this is the simplest to implement. However, it does ignore the effect of multiple rules with a similar outcome, as the maximum is always taken. This is not a desirable outcome as the expectation is that the end result should trend towards the majority. If sum-min inference is used then the outputs will be added and all of the rules will contribute independently.

The inference method was implemented in a C program and tested and verified using MATLAB. To avoid using floats, truth values were represented as integers between 0 and 100. While this introduced some rounding errors these did not exceed one percent.

TABLE I. FUZZY LOGIC RULES FOR OBSTACLE-AVOIDANCE AND NAVIGATION

Fuzzy Logic Rules	
IF Far Left Distance is Close	THEN Steering is Soft Right
IF Far Right Distance is Close	THEN Steering is Soft Left
IF Left Distance is Close	THEN Steering is Hard Right
IF Right Distance is Close	THEN Steering is Hard Left
IF Left Distance is Near	THEN Steering is Soft Right
IF Right Distance is Near	THEN Steering is Soft Left
IF Heading is Far Left AND Far Left Distance is Far	THEN Steering is Hard Left
IF Heading is Left AND Left Distance is Far	THEN Steering is Soft Left
IF Heading is Centre AND Centre Distance is Far	THEN Steering is Centre
IF Heading is Right AND Right Distance is Far	THEN Steering is Soft Right
IF Heading is Far Right AND Far Right Distance is Far	THEN Steering is Hard Right
IF Heading is Far Left AND Far Left Distance is Near	THEN Steering is Soft Left
IF Heading is Left AND Left Distance is Near	THEN Steering is Soft Left
IF Heading is Right AND Right Distance is Near	THEN Steering is Soft Right
IF Heading is Far Right AND Far Right Distance is Near	THEN Steering is Soft Right
IF Centre Distance is Close AND Right Distance is Close AND Left Distance is NOT Close	THEN Steering is Hard Left
IF Centre Distance is Close AND Left Distance is Close AND Right Distance is NOT Close	THEN Steering is Hard Right
IF Centre Distance is Near AND Right Distance is Close AND Left Distance is NOT Close	THEN Steering is Soft Left
IF Centre Distance is Near AND Left Distance is Close AND Right Distance is NOT Close	THEN Steering is Soft Right
IF Centre Distance is Close AND Left Distance is Close AND Right Distance is Close AND Far Left Distance is Close AND Far Right Distance is NOT Close	THEN Steering is Hard Right
IF Centre Distance is Close AND Left Distance is Close AND Right Distance is Close AND Far Right Distance is Close	THEN Steering is Hard Left

The bisector method was used for defuzzification. This is faster than finding the centroid as it can be done in linear time using only addition operations and produces similar results [36]. The bisector method involves calculating the point where the area is bisected. This was achieved by using two pointers at the start and end of the output range with associated area values. The pointer with the smallest area is moved towards the other side and the area is incremented by the current value. This is repeated until the pointers overlap and that position is used as the bisector.

VI. RESULTS

A. MATLAB Simulation

As the fuzzy logic control system was originally developed and tested in MATLAB, this tool was used as simulator for the whole system. A rudimentary script was created for this purpose. The script used MATLAB's built-in fuzzy logic functionality, and modeled the infrared sensors using ray tracing. The sensors were assumed to have a one unit-wide beam for simplicity. Black and white images were imported and used as maps. White was used to represent open space and black was interpreted as an obstacle. Figure 7 shows the simulated path for one of the testing scenarios.

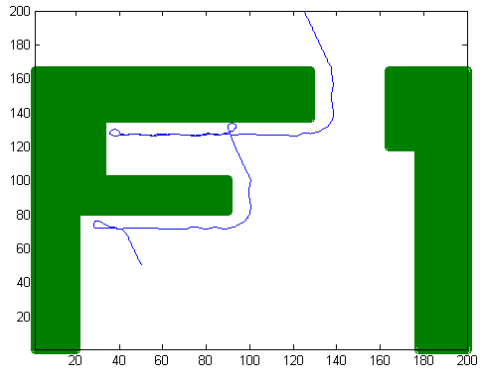

Figure 7. Navigation and obstacle-avoidance simulation results

Although the simulator does not use an accurate model of the physical robot; it provided a good visualization for the fine-tuning of the control system. A number of test maps were used to evaluate performance in particular scenarios such as U-shaped obstacles, corridors and dead-ends.

B. Real-World Performance

A backyard, local school and parking lot were used as test environments for the platform. During the tests the small vehicle successfully navigated from one set of GPS co-ordinates to another, negotiating different types of terrains and obstacles. Figure 8 shows one of the runs when the robot had to find its way from an indoors environment to a destination across the street. The straight line represents the direct path between origin and destination points, while the curved line represents the actual path followed by the robot. With no map or prior information the destination was reached without any collisions. The same result was obtained in every test.

Figure 8. Demonstration of autonomous navigation

VII. Conclusions and Future Work

The development and test of an autonomous robotics platform for educational purposes has been presented. All requirements and specifications were satisfied. At $755 USD, the platform is affordable for use in undergraduate projects. All source code, parts-list and developments tools are freely available on request. This is a strong incentive for academic adoption as licensing costs are non-existent. It is also the only platform to provide proven built-in obstacle avoidance and navigation. The platform may significantly accelerate the development of mobile robots as a significant portion of common and underlying functionality is already implemented and tested. However it is important to note that at this stage the platform is still a prototype.

There are two major avenues available for future work. One is to extend the existing functionality with more hardware and software modules. Cliff detection sensors could be placed on the front barrier to prevent the platform from going over a steep edge. Wireless telemetry could also be incorporated, allowing for position and sensor data to be reported and waypoints to be set remotely. It is expected that all these tasks can be achieved within the $1,000 USD limit.

The other avenue involves improving the platform from an educational point of view. Currently, the use of a relatively low level programming language restricts its use to academics and university students. In order to open up the platform to a wider audience, user-friendly development tools need to be created. The whole system could be modeled as a diagram of interconnected blocks which can be visualized and also simulated. In addition, a wireless link to the physical platform could provide real-time feedback and control for debugging and learning purposes.

Even in its current form, Curtin's robotics development platform is a highly capable tool that is expected to be used as the base for several robots in the future.

References

[1] The LEGO Group. (2009) LEGO Mindstorms NXT. [Online]. http://mindstorms.lego.com/Products/Default.aspx.

[2] P. Lewis. (2007) Frequently Asked Questions about UAVs. [Online]. http://www.uavforum.com/library/librarian.htm.

[3] Surveyor Corporation. (2008) Surveyor Robotics Journal. [Online]. http://www.surveyor.com/cgi-bin/robot_journal.cgi/2008/08/.

[4] M. Hennerich. (2006, Apr.) uClinux on the Blackfin DSP Architecture: Part 3. [Online]. http://www.eetimes.com/esc/showArticle.jhtml?articleID=185302712.

[5] University of Notre Dame . (2008, Dec.) From Conventional to Intelligent Control. [Online]. http://www.nd.edu/~isis/history.html.

[6] K. Passino and S. Yurkovich, Fuzzy Control. California: Addison Wesley Longman, 1998.

[7] P. Norvig and S. J. Russell, Artificial Intelligence: A Modern Approach, 2nd ed. Alexandria, VA: Prentice Hall, 2002.

[8] M. Veloso and A. Costa. MAPPEL - Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning. [Online]. http://www.lti.pcs.usp.br/mappel/nsf/nsf.html.

[9] M. Singh and D. Parhi, "Intelligent Neuro-Controller for Navigation of Mobile Robot," in International Conference on Advances in Computing, Comm. and Control (ICAC3'09), 2009, pp. 123-128.

[10] K. Passino and S. Yurkovich, Fuzzy Control. California: Addison Wesley Longman, 1998.

[11] O. Henlich. (1997, May) An Overview of Local/Personal Robot Navigation. [Online]. http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/oh/.

[12] N. Bowditch, The American Practical Navigator. Bethesda, Maryland: National Imagery and Mapping Agency, 1995.

[13] Y. Fuke and E. Krotkov, "Dead Reckoning for a Lunar Rover om Uneven Terrain," in Procs of the 1996 IEEE International Conference on Robotics and Automation , Minneapolis, 1996, pp. 411-416.

[14] H. Weinberg and C. Lemaire. (1997) Using the ADXL202 Accelerometer as a Multifunction Sensor in Car Alarms. [Online]. http://www.analog.com/static/imported-files/application_notes/5032436457109743495432152849573 0car_app.pdf.

[15] M. Amundson. (2006) Dead Reckoning for Consumer Electronics. [Online]. http://www.ssec.honeywell.com/magnetic/datasheets/Dead_Reckoning_Consumer_Electronics.pdf.

[16] Lynxmotion Inc. (2009) Lynxmotion Robot Kits. [Online]. http://www.lynxmotion.com/.

[17] Imagecraft . (2009, Jan.) Parallax Propeller C development tools. [Online]. http://www.imagecraft.com/devtools_Propeller.html.

[18] Arduino. (2009) Arduino Diecimila. [Online]. http://arduino.cc/en/Main/ArduinoBoardDiecimila.

[19] Pololu Corporation. (2009) Orangutan X2. [Online]. http://www.pololu.com/catalog/product/718.

[20] Acroname Robotics. (2008, Jan.) Sharp IR Rangers. [Online]. http://www.acroname.com/robotics/info/articles/sharp/sharp.html.

[21] USGlobalSat. (2009) EM-406A (SiRF III). [Online]. http://www.usglobalsat.com/p-46-em-406a-sirf-iii.aspx

[22] SiRF. (2004, Nov.) SiRFstarIII™ GPS Single Chip - A High Performance GPS in a Small Form Factor. [Online]. http://uk.ts.fujitsu.com/rl/servicesupport/techsupport/pda/General/SiRFstarIIIGSCf.pdf

[23] Sparkfun Electronics. (2009, Mar.) GPS Buying Guide. [Online]. http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=127

[24] Oxford Technical Solutions. (2009) Glossary of Terms. [Online]. http://www.oxts.com/default.asp?pageRef=40

[25] Honeywell. (2006, Jan.) Digital Compass Solution HMC6352. [Online]. http://www.sparkfun.com/datasheets/Components/HMC6352.pdf

[26] Robot Electronics. (2007, Mar.) CMPS03 - Compass Module. [Online]. http://www.robot-electronics.co.uk/htm/cmps3tech.htm

[27] I. Buchmann. (2005) What's the best battery?. [Online]. http://www.batteryuniversity.com/partone-3.htm

[28] M. Thompson. (2001) Generic battery technology comparison. [Online]. http://www.madkatz.com/ev/batteryTechnologyComparison.html

[29] SmartGauge Electronics. (2008) An in depth analysis of the maths behind Peukert's Equation. [Online]. http://www.smartgauge.co.uk/peukert_depth.html

[30] R. Cringely. (2006) Safety Last. [Online]. http://www.nytimes.com/2006/09/01/opinion/01cringely.html?_r=1

[31] Atmel. (2002, Sep.) AVR Studio - Software Development Environment. [Online]. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[32] WinAVR. (2009) WinAVR: AVR-GCC for Windows. [Online]. http://winavr.sourceforge.net/

[33] Real Time Engineers. (2009) The FreeRTOS Project. [Online]. http://www.freertos.org/

[34] M. Dalheimer. (2008) FreeRTOS for ATMega644. [Online]. http://gonium.net/md/2008/08/10/freertos_for_atmega644/

[35] L. Barello. (2007) Interfacing with GP2D02 sensors. [Online]. http://www.barello.net/Papers/GP2D02/

J. Rajewski. (2008) GP2Y0A2YK0F Sharp Distance Sensor. [Online]. http://www.societyofrobots.com/robotforum/index.php?topic=2712.30