LEVERAGING MOBILE DEVICES TO ENHANCE THE PERFORMANCE AND EASE OF PROGRAMMING FOR LOW-COST MOBILE ROBOTS

Ryan P. Grainger and David I. Grow

Department of Mechanical Engineering, New Mexico Tech, Socorro, New Mexico, USA

ABSTRACT

Programming simple robots allows teachers to reinforce unified science, technology, engineering, and math (STEM) concepts. However, for many educators, the cost and computer requirements for robotics kits are prohibitive. As mobile devices have become increasingly ubiquitous, low cost, and powerful, they may prove to be an attractive means of coding for, controlling, and enhancing the capabilities of low-cost mobile robots. This study looks into the viability of using LEGO Mindstorms NXT and Google Android devices by using Bluetooth to establish a link between the two. This allows for the exchange of live data remotely for use in various applications with the hope of creating a low-cost mobile programming environment. The mobile applications developed were able to successfully exchange data with NXT hardware via Bluetooth and show evidence that mobile devices can be used as a tool to assist in robotic programming in education.

KEYWORDS

Android, Application Programming Interfaces, Bluetooth, Educational Robots, & LEGO Mindstorms NXT

1.INTRODUCTION

The field of Robotics continues to grow significantly, has entered into nearly every part of industry, and global demand is at an all-time high [1]. However, for those with little or no programming experience, the effort required to learn even basic reprogramming is often prohibitive. This issue is compounded by the inconsistency in the K-12 programming emphasis. Industry has made strides in addressing this gap by creating abstracted, and even graphical, programming environments such as LabVIEW [2]. Nonetheless, many of these environments have limitations including:

- Need for modestly advanced computer hardware
- Need for manual software/package updates
- Proprietary file types requiring external conversion
- Interaction limited to the machine the software is installed
- Restricted support for additional hardware components
- Costly licensing fees

Mobile devices such as phones and tablets, which are increasingly entering the educational system, address these limitations and offer an alternative for robotic programming. Additionally, mobile devices' onboard sensors and communications can be used to improve the robot's basic abilities. Furthermore, programming on a student's own mobile phone will promote a sense of familiarity, which could increase student engagement, and also allow school funds to be spent in other areas.

DOI :10.5121/ijite.2014.3301

To explore this concept, a virtual instrumentation panel, motor calibration tool, and advanced instrumentation panel were developed as a proof-of-concept to show how a mobile device can display information for a robot along with its sensors and to improve a robot's sensing abilities to help with problem solving. Here we provide evidence that the use of a mobile device yields multiple benefits over the traditional method and provides guidance for future work in this area. The specific contributions of this paper are:

1. A real-time stream of information between a mobile device and a robot.
2. An easy-to-use application to determine and improve a robot's directional commands.
3. An exploration into the feasibility of creating a fully interactive mobile programming application, highlighting remaining challenges but demonstrating that this approach is feasible in the near-term.

For the purposes of experimentation, this research uses the LEGO Mindstorms NXT, a low-cost and versatile robotics kit, and a Google Android Device in configurations similar to those in Figure 1.

Figure 1. Setup (NXT and Android separate, NXT and Android integrated). Mobile device can be used with the robot separately to command remotely and display data to the user, or integrated to allow sensors from the mobile device to advance the robot's capabilities.

2.RELATED WORK

Alternative programming environments for the NXT have been studied to advance usability and hardware capabilities. In 2007, Kim and Jeon [3] compared the standard NXT-G software, powered by LabVIEW of National Instruments, with that of Microsoft's Robotics Developer Studio. They determined that while a visual programming environment is the best way to reach a larger audience, a more flexible environment is needed for experienced programmers. Wadoo and Jain [4] determined in 2011 that ROBOTC could advance the abilities of the NXT using built-in functions for PID control to develop a control systems laboratory at a minimal cost, in comparison to expensive equipment that would have been difficult to program and provide limited use. These insights push development of new programming environments to remain graphical while also being intuitive to both new and old users

There is much debate over whether to use a graphical or textual programming environment. In 2008, Azemi and Pauley [5] performed a course-study involving the shared use of C++ and MATLAB; however, while both are textual-based programming languages, MATLAB was found by students to have a more intuitive design as there were many graphical-based components to

assist in programming. Additionally, the GUI allowed for better understanding of major topics that were discussed during the course. The main issue was that MATLAB had to be purchased and installed on their own personal computers. With their 2006 survey, Yoder and Black [6] stated that the question "Which is better?" may never be answered; they found that students actually preferred using a graphical-based language (LabVIEW). As robotics in education has grown, the need for an intuitive way of programming is required.

Papers pertaining to robotics in education referenced the STEM disciplines (Science, Technology, Engineering, and Mathematics). Ekong, Choi, and Rascoe [7], in 2010, successfully introduced robotics into a middle-school STEM curriculum to inspire students to study and pursue careers in science and engineering. The workshops involved teachers using a prior version of the LEGO Mindstorms kit with the biggest difficulty being a lack of an intuitive programming environment. In 2012, Saygin, et. al., [8] determined that the use of robotics in an educational environment helped engage students to become active learners and presented engineering concepts in concrete, relevant, and real-world contexts.

Based on prior work, it appears that a GUI interface is the most productive for students during the learning process. Increasing use and access would allow students to engage in programming more frequently. This research looks into the potential of using mobile devices as a new solution.

3.METHODS

Virtual instrumentation for the PC has been rapidly adopted in the past 20 years. They consist of an industry-standard computer or workstation with powerful application software, cost-effective plug-in boards, and driver software [9]. However, for average schools, these development boards are classed as high-level equipment that could not be purchased on a budget, and the same goes for the software and industry-standard computers. Since many schools already have access to the LEGO Mindstorms NXT kits, the need to purchase additional development boards would be removed; furthermore, having students use their own low-cost mobile devices would reduce the need to purchase up-to-date computers. With this in mind, a greater focus has been placed on a mutual communication standard and software development.

3.1 Bluetooth

Bluetooth is a key integration component for this research, as it allows the exchange of data over short distances between various devices. Since the LEGO Mindstorms NXT and most mobile devices come with Bluetooth, it allows for wireless communication of sensor data and motor commands. The NXT's Bluetooth is based on a CSR BlueCore^{TM} 4 v2.0 +EDR System, supporting the Serial Port Profile (SPP) [10]. This profile is based on RFCOMM protocol and emulates a serial cable to provide a simple substitute for the existing RS-232 protocol [11] (the original intention of Bluetooth [12]), including the familiar control signals. All Google Android devices support this; however Apple officially lacks support for this on its iOS devices. For this reason, and its worldwide popularity, Google Android was chosen.

3.2 MIT App Inventor

The basic applications (Virtual Instrumentation and Motor Calibration) were developed using "MIT App Inventor," originally created by Google under the name "App Inventor for Android." It is now maintained by the Massachusetts Institute of Technology (MIT) [13]. This development software was chosen not only because it is a fast and efficient application development tool but because the LEGO Mindstorms NXT components are already built in [14]. App Inventor allows the development of applications for Android phones using a click-and-drag web-based interface

and either a connected mobile device or an emulator. Developing apps appear on the device, step-by-step, as pieces are added. This allows application testing while building. When the application is finished, it can be packaged to produce a stand-alone application to install.

3.3 Xamarin Studio and MonoBrick

Xamarin Studio is an Integrated Development Environment (IDE) for cross-platform mobile development [15] that allows development in C# Language with access to the .NET Framework [16]. MonoBrick is a LEGO Mindstorms communication library, developed by Anders Søborg, written as a plug-in for Xamarin Studio. The library allows mobile devices to communicate with the LEGO Mindstorms NXT brick using Bluetooth. MonoBrick has the following features [17]:

- Support for more than 20 analog and I^2C sensors
- Individual and vehicle motor control
- A send-and-receive mailbox system
- The ability to set the brick name, acquire battery level, read firmware version, etc.
- Play tones and sound files
- Use the onboard file system to download and upload files
- Start and stop on-brick programs
- Use exceptions to catch sensor and connection errors
- Open and close connections with multiple NXT units

This environment and library will allow for better applications to be developed in comparison to those made with MIT App Inventor. After contacting Søborg, the source code for MonoBrick was acquired to expand the library and allow for custom sensors.

4.RESULTS

4.1 Real-time Data Acquisition and Control

With the built in functions of MIT App Inventor, a basic Virtual Instrumentation tool was made (Figure 2). This tool provides the ability to connect the Mobile Device and NXT via Bluetooth, display sensor and motor tachometer values in real-time, and allow the user to set different sensor types and motor positions. A button located at the top is used to connect/disconnect the NXT with the mobile device, with the status of connection shown on the button text. Once a connection is made, the sensor and motor buttons can be pressed to change various settings. The four sensor buttons correspond to the four ports located on the NXT; each sensor has the selective options of:

- Light
- Sound
- Touch
- Ultrasonic

Once a port is set to the correct sensor, the mobile device is able to acquire data in real-time. The motors (A, B, & C) can also be read individually, however there is an additional option to set a motor position using the arrow buttons to the left and right of each motor, incremented by ±90 degrees. The addition of motor control resulted in a closed-loop control architecture to be employed, primarily due to small amounts of lag between actual and displayed motor positions. It was also found that with the addition of each sensor and motor giving real-time information, the lag increased, causing stability issues. This is believed to be due to the serial nature of the

Bluetooth connection. Querying and streaming data from multiple sensors and motors moves beyond the limitations of Bluetooth.

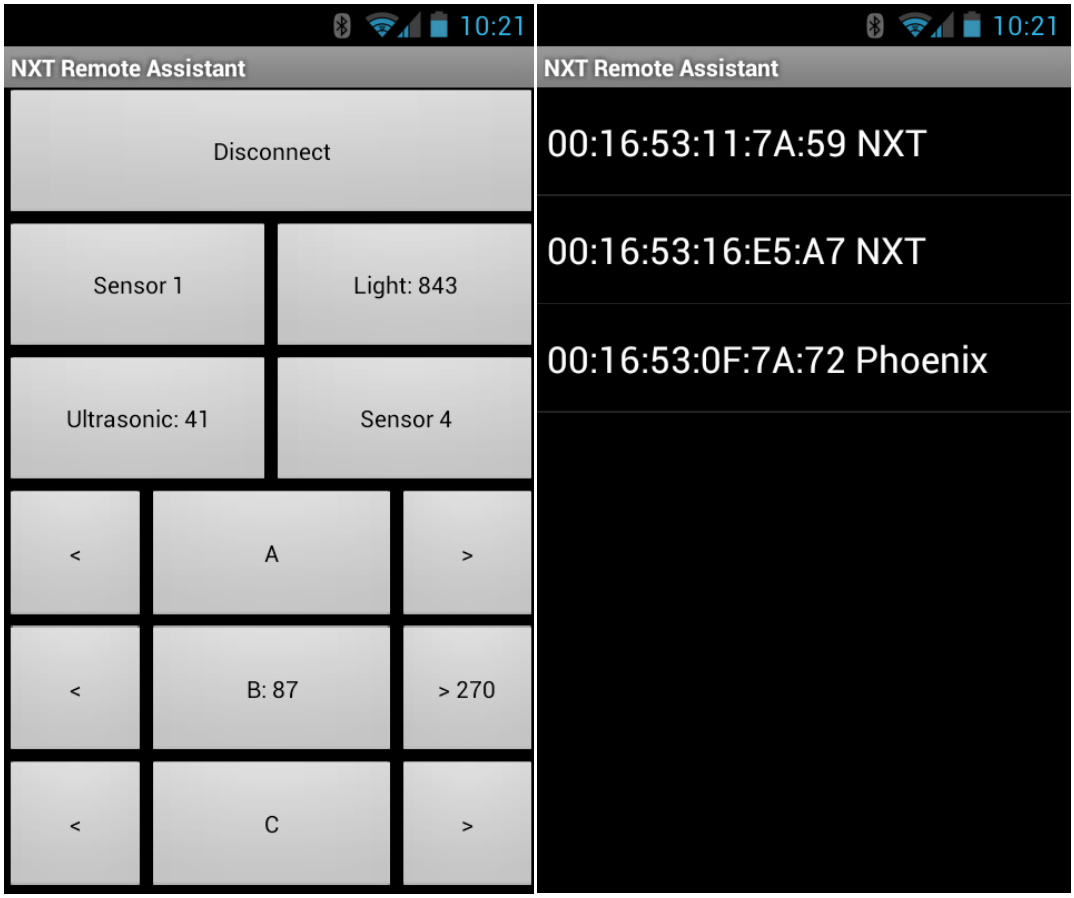

Figure 2. Entire Virtual Instrumentation Application (Main Page, Bluetooth List). Allows user to initialize a connection, select and observe sensor/motor readings, and command motor position.

Two tractable solutions were considered. First, to change to a different connection, such as USB, which all NXT's and most Android devices have. This connection would require additional hardware and for the mobile device to support it. Second, the more feasible option was to develop a system that prioritizes certain readings. If motor position is absolutely critical it can be set at a higher priority than that of a touch sensor. For this research the sensors were set, by default, to not stream data unless requested.

The results showed that the mobile application was able to successfully poll data from the NXT hardware via Bluetooth. It was found that a limitation of the Bluetooth serial connection exists and will need to be improved in future work with this method.

4.2 Interaction with Automated Routines

During programming, NXT motors are given a set direction that is determined as either forward or backwards. This setup works with the robot configuration in the provided manual; however not all robot configurations have the motors in the same orientation, and it becomes difficult to work out which motor directions will allow the robot to move as a whole. The only way to determine motor direction is to either guess and test, or to manually determine which motors are

configured to which ports, motor orientation, and the effect of any present gearing systems. Both of these solutions work, but can be difficult and cumbersome.

The primary issue is that the robot does not know its actual orientation and configuration, which is difficult when the robot has minimal sensing of its surroundings. Our solution was to use a mobile device's various sensors and link the robot to them through Bluetooth. This allows the mobile device to calibrate the motors and determine the direction each motor needs to move to achieve a specific movement. This calibration process will specifically involve the NXT motors, the mobile device's compass, and a Bluetooth connection. The mobile device will run the motors through a process of directional movements and determine how the robot acts with the onboard sensors; the result will then be displayed so the user can program the robot correctly. The developed environment is capable of:

1. Determining the forward direction of the robot
2. Obtaining motor location and orientation on the chassis
3. Displaying which direction the motors must be set to achieve a specified velocity

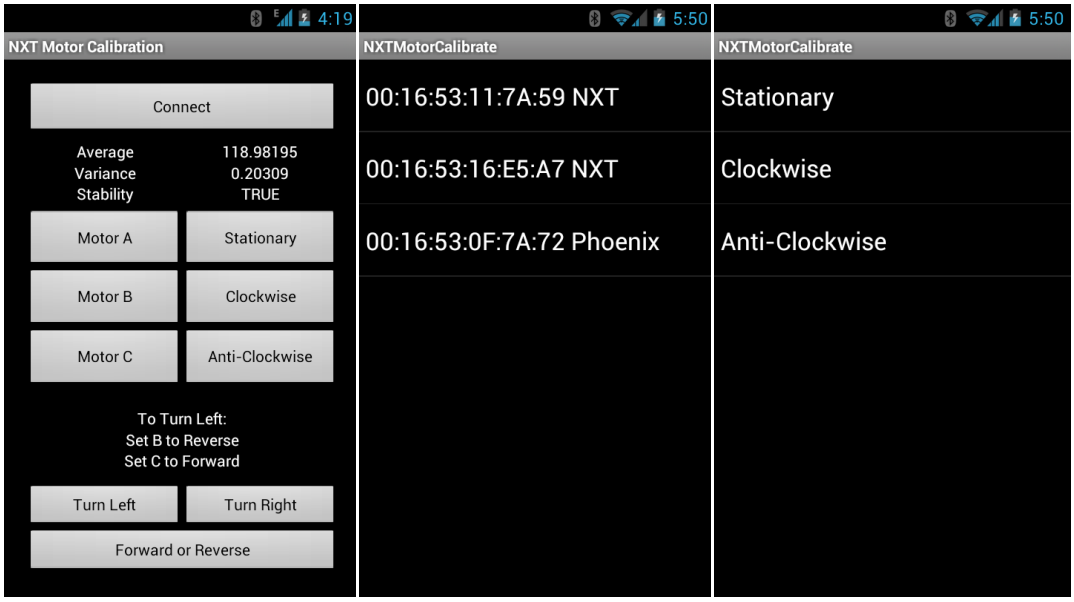

Figure 3. Entire Motor Calibration Application (Main Page, Bluetooth List, Direction List). Determines motor configuration with the use of compass readings and user input.

The Motor Calibration Tool, seen in Figure 3, was designed to be displayed as a Single-Page Application/Interface (SPA/SPI), because the user would usually be required to observe all data at one given time, with the exception of Bluetooth connection and manual overrides for selected tests. A button is again used to connect/disconnect the NXT to the mobile device; the status of the connection is shown by the button text. An active readout is always displayed for the magnetometer readings The average reading is based on a ten-wide running average, the variance reading is based on the standard deviation of the running average, and the stability readout reads as true when the deviation is less than 0.5 degrees. The motor control buttons (A, B, & C) perform forward motor commands and read the average reading both before and after the movement is performed to determine the direction of rotation. The buttons adjacent to the motor control buttons allow for the user to manually override the direction of rotation, using a list, to one of the three choices: Stationary, Clockwise, and Anti-Clockwise. The lower half of the

display either shows a message that signifies too few/many motors being used or an output for the two correctly selected and calibrated motors. When the tests are successful a command example is given for a left turn along with left and right movement tests and a command for either forward/backward movement.

In summary, the application is able to successfully determine motor directions for use in programming and the manual override assisted in any incorrect calibrations. Magnetic interference proved to be the largest issue. Both the environment and the robot itself caused issues with the magnetometer; thus the most successful tests were performed when the robot was used on flat ground with no walls or objects in the surrounding area, with the device placed farthest away from the motors and battery pack.

4.3 Augmented Control and Processing

Development with MIT App Inventor proved difficult for tasks beyond the basic back-and-forth communication and simple routines. This required switching to Xamarin Studio and with it a code-based programming environment. There are numerous advantages to using Xamarin compared to MIT App Inventor:

- Variety of screen sizes, which allows the overall appearance of the application to have a more professional appearance
- Multiple windows, to retain information, with the ability for data to be transferred between them, allowing for a more immersive environment
- Wider range of layouts & views; preconfigured menu systems require less time to be spent on the appearance and for the user to become more familiar with standardized layouts
- C# Support, allowing for cross-platform code (iOS, Android, Windows, and Mac)
- Smaller file size, allowing the developed application to take up less space on a device
- No application size restriction, meaning no application becoming too big for the development system to handle
- Support for older and newer OS versions, allowing all possible users the ability to run software without incompatibility issues
- Ability to read an assortment of onboard sensors, including those appearing on future devices

A further advantage of the MonoBrick library is enhanced communication between Android and the NXT, allowing support for more than 20 analog and I^2C third-party sensors. Moreover, with access to source code, many more drivers can be developed, allowing for a large database of NXT sensors.

With the functions of Xamarin and MonoBrick, an Advanced Virtual Instrumentation tool was made (Figure 4). This tool provides the ability to connect the Mobile Device and NXT via Bluetooth, display motor tachometer values, NXT sensor readings, and Android sensors in real-time. Since Xamarin allows multiple windows, the connection settings have been moved to the initial screen, requiring an established connection to be made before anything else can be done. Once a connection is made, the sensor ports can be set to connect to various types of sensors and to selectively interpret the inputs as boolean, scaled, percentage, and raw values. These are chosen using an implementation similar to a drop-down menu.

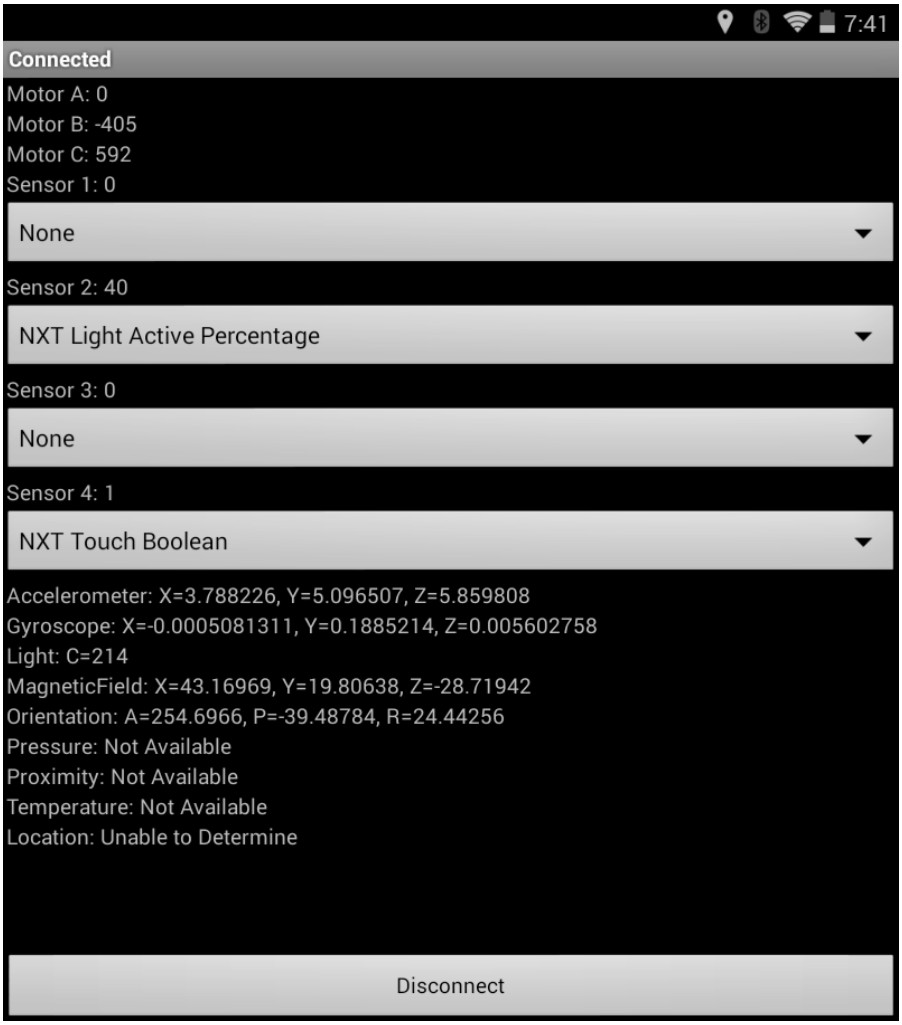

Figure 4. Entire Advanced Virtual Instrumentation Application. Allows user to initialize a connection, observe motor tachometer readings, and display sensor readings from both the robot and mobile device.

The application is able to determine if an onboard sensor exists, and constantly displays all data relating to the sensor. Those not available, or those unable to be read, inform the user of such. These onboard sensors are run in a separate subroutine as they can be read faster than those from the NXT. Sensors such as GPS normally can only be used outside with clear access to satellites; however there are a variety of ways to determine location depending on what additional equipment the device has. For example, mobile phones can use localization of radio towers to give a rough estimate of position.

An added bonus of Xamarin being a code-based programming environment is that programs can be run directly inside of the application allowing for quick tests and validations. Supported tests include:

- Real-time PID tuning during line-following task
- PID control of motor angle trajectories
- Android compass headings to determine robot direction
- Point-to-point GPS navigation and path-finding

These tests demonstrate some of what is possible with a Graphical Programming Environment on an Android Device.

4.4 Graphical User Interface

Using the knowledge gained, a preliminary Graphical User Interface was developed (Figure 5) that allows the user to add commands and graphically alter the settings of various conditions and actions that the robot can perform. When a command is accessed various visual inputs are displayed to allow command settings to be changed, a dynamic onscreen keyboard is shown when needed; if a Bluetooth connection has already been established with an NXT, live feedback from sensors and motors are displayed allowing the user to make appropriate modifications. This enables the user determine what the robot does with sensor data and perform actions accordingly. The list of programming commands makes use of an indent style to convey the program's structure; similar to that of a C programming language or its descendants, with start and end symbols to represent braces. The indentation allows for control flow constructs such as conditions or loops to easily identify by the user, in addition various colors and shapes are applied to visually determine the type of command used. While only in the initial stages of development, it is clear that a graphical representation of programming will allow users to easily transition, when necessary, into a text-based language with little difficulty.

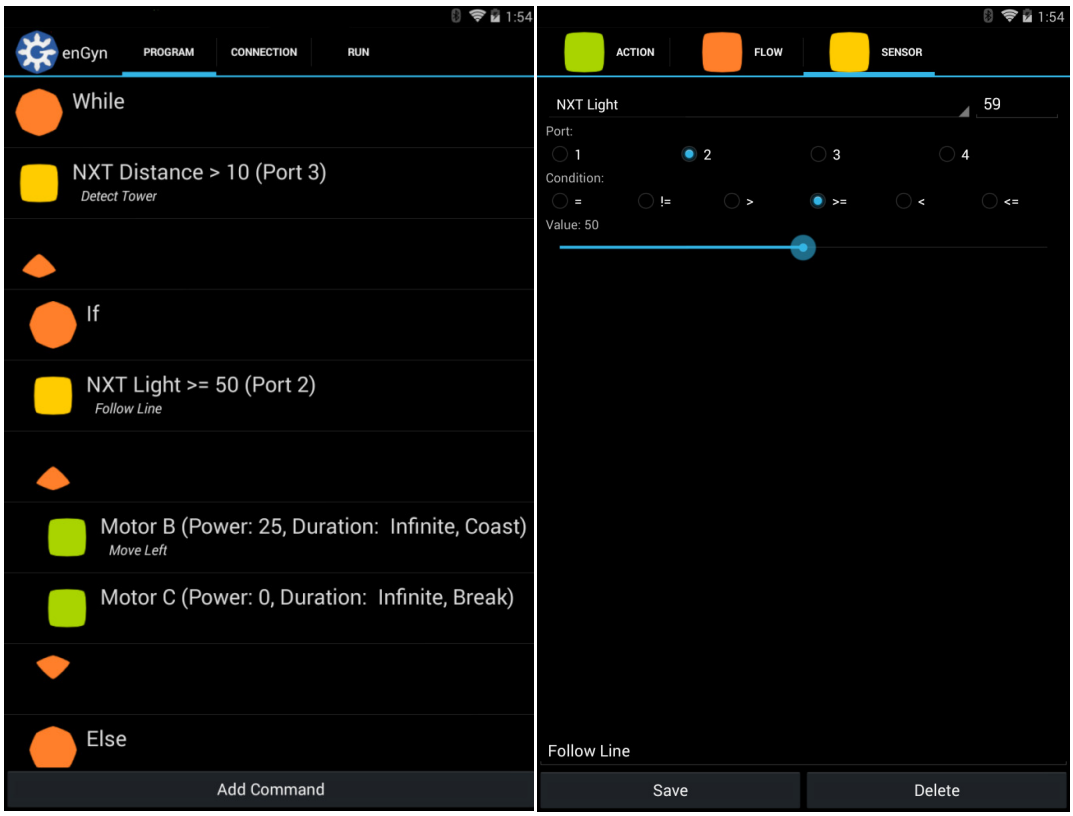

Figure 5. Preliminary Graphical User Interface (Programming Window, Command Window). Allows the user to create a list of commands using a dynamic interface that the robot can then perform.

4.5 Interpreter

With the graphical structure in place the next step was to turn the user created commands into working commands that the Android device could interpret and run. A database structure was created using SQLite which allowed for fast and easy access to vital information among operations. While SQLite's read operations can be multitasked, writes can only be performed sequentially; this limitation is ideal as writing to the file would only occur during the programming process but reading is performed throughout the application; and because of its small size, SQLite is well suited to embedded systems like Android.

The creation of a database was only part of the solution, the commands had to be interpreted along with the implementation of loops to simplify development, a basic algorithm was developed in MATLAB to separate issues with Android and provide a simpler language with which to debug with. Several parts were carried out using MATLAB; these parts were mainly related to command components that were not already built using Xamarin Studio. These components involve the implementation of a "goto" or "jump" functions to create "while" loops and "if-else" statements; as well as their respective start and end braces. Once the MATLAB script was reintegrated into C# the initial testing showed positive results and basic programs were able to created easily and perform the set tasks correctly.

5.DISCUSSION & FUTURE WORK

A key broader impact of this work is that the same programming environment can be utilized in primary education to help in assuring that a greater number of next-generation engineers can have an early introduction to programming. Most schools focus on teaching students how to use a computer and run available applications, rather than exploring the deeper concepts such as computational problem-solving, which lay the foundation for innovation [18]. In both K-12 and many industries unfamiliar with programming, an intuitive programming interface is needed, especially where large numbers of low cost, yet highly capable, mobile robots are present.

The present study provides preliminary evidence that mobile devices are a great tool to assist in programming robots for educational use. With the growing issue of obsoletion caused by rapidly evolving technologies, this application is a promising and affordable solution for students, educators, and other users of robotic systems. Future work will allow greater advancement in application complexity and we are working to support additional sensors, improving filtering, and improving polling techniques from sensors and motors. The application developed in the research is currently in the public beta-test stage and will be available in the Google Play store in the near future.

With the range of built-in sensors in mobile devices constantly increasing, an application that can combine these with a mobile robotic platform will allow low-cost robotics to have advanced capabilities. A fully interactive mobile programming application could be developed to run on mobile devices, freeing robotic development to be used in mobile environments. The use of mobile devices as programming tools would allow on-the-go development without the difficulty of using large machines, such as laptops which cannot be integrated into a robotic platform. Networked mobile devices could allow users to exchange programs and work together simultaneously, even when not working in the same room.

REFERENCES

[1] IFR: All-time-high for industrial robots in 2013. [Online]. Available: http://www.ifr.org/news/ifr-press-release/ifr-all-time-high-for-industrial-robots-in-2013-601/

[2] National Instruments LabVIEW. [Online]. Available: http://www.ni.com/labview/

[3] S.-H. Kim and J. W. Jeon, (2007) "Programming LEGO Mindstorms NXT with visual programming," in Control, Automation and Systems, 2007. ICCAS '07. International Conference on, Oct 2007, pp. 2468–2472.

[4] S. Wadoo and R. Jain, (2012) "A LEGO based undergraduate control systems laboratory," in Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island, May 2012, pp. 1–6.

[5] A. Azemi and L. Pauley, (2008) "Teaching the introductory computer programming course for engineers using Matlab," in Frontiers in Education Conference, 2008. FIE 2008. 38th Annual, Oct 2008, pp. T3B–1–T3B–23.

[6] M. Yoder and B. Black, (2006) "Work in progress: A study of graphical vs. textual programming for dsp," in Frontiers in Education Conference, 36th Annual, Oct 2006, pp. 17–18.

[7] D. U. Ekong, T. A. Choi, and B. Rascoe, (2011) "A Robotics Workshop for Middle School STEM Teachers", in proceedings of ASEE SE Section Annual Conference, Charleston, SC. April 10-12, 2011.

[8] Saygin, C., Yuen, T., Shipley, H., Wan, H., & Akopian, D, (2012) "Design, development, and implementation of educational robotics activities for K-12 students" 2012 ASEE Annual Conference.

[9] National Instruments virtual instrumentation. [Online]. Available: http://www.ni.com/white-paper/4752/en/

[10] LEGO Mindstorms NXT Bluetooth developer kit. [Online]. Available: http://mindstorms.lego.com/en-us/support/files/default.aspx

[11] Bluetooth basics. [Online]. Available: http://www.bluetooth.com/Pages/Basics.aspx

[12] Bluetooth fast-facts. [Online]. Available: http://www.bluetooth.com/Pages/Fast-Facts.aspx

[13] What is App Inventor? [Online]. Available: http://appinventor.mit.edu/explore/content/what-app-inventor.html

[14] LEGO Mindstorms components. [Online]. Available: http://appinventor.mit.edu/explore/content/legomindstorms.html

[15] Xamarin Studio. [Online]. Available: http://xamarin.com/how-it-works

[16] C# and .NET. [Online]. Available: http://msdn.microsoft.com/en-us/library/vstudio/z1zx9t92.aspx

[17] MonoBrick. [Online]. Available: http://www.MonoBrick.dk/software/MonoBrick/

[18] C. e. A. Wilson, (2010) "Running on empty: The failure to teach K-12 computer science in the digital age". [Online]. Available: http://www.acm.org/runningonempty/

Authors

Ryan Paul Grainger received the B.Sc. degree in mechanical engineering and the M.Sc. degree in mechatronics from New Mexico Institute of Mining and Technology, Socorro, New Mexico, USA, in 2012 and 2014, respectively. Since then, he has been with Inquiry Facilitators, Bernalillo, New Mexico, USA, where he is currently a facilitator & consulting engineer.

David Issac Grow received the B.Sc. degree in physics and the M.Sc. degree in mechanical engineering from University of Utah, Salt Lake City, Utah, USA, in 2004 and 2006, respectively and the Ph.D. degree in mechanical engineering from Johns Hopkins University, Baltimore, Maryland, USA, in 2011. Since then, he has been at New Mexico Institute of Mining and Technology, Socorro, New Mexico, USA, where he is currently an assistant professor of mechanical engineering.