2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology

SiFEB – A Simple, Interactive and Extensible Robot Playmate for Kids

Hashini Senaratne[1], Pubudu Gunatilaka[2], Udith Gunaratna[3], Yasura Vithana[4] and Chathura de Silva[5]
Department of Computer Science and Engineering
University of Moratuwa
Moratuwa, Sri Lanka
[1]hashini.10@cse.mrt.ac.lk, [2]pubudu.10@cse.mrt.ac.lk,
[3]udith.10@cse.mrt.ac.lk, [4]yasura.10@cse.mrt.ac.lk,
[5]chathura@cse.mrt.ac.lk

Piyum Fernando[6]
Singapore University of Technology and Design
Singapore
[6]fernando@sutd.edu.sg

***Abstract*—In this paper we present work in progress on SiFEB, a simple, interactive and extensible robot playmate for kids intended to nurture the engagement in STEM (Science, Technology, Engineering and Mathematics) related activities through playing. SiFEB's simplified and interactive hardware and software modules present users without prior experience, with the ability to build robots without going deep in either aspect. Kids can build robots of their interest using attachable hardware modules and related programming components using an easy to understand visual programming language that resembles natural commands. The potential developers may utilize the framework provided, to build hardware and software components to the system expanding the existing capabilities. Preliminary results from the prototypes show that SiFEB has the potential of being an effective learning platform for young children in STEM related areas.**

Keywords – children; robotics; STEM; visual programming

I. INTRODUCTION

Robotics is a rapidly developing field that encompasses emerging technologies to build automated machinery which take the place of humans. Although robots were present only in highly industrial applications few decades ago, they have now started to mark their presence very strongly in some domestic applications such as hobby projects and games as well [1, 2].

From recent researches of different engineer associations such as VIK (Flemish Association of Engineers), it has been identified that there is a significant drop in STEM-related (Science, Technology, Engineering and Mathematics) knowledge of the students [3]. It has been also figured out that Maker Movements and DIY (Do It Yourself) trends that are closely related with the activities like building and programming robots, can impact the above problem positively [3]. Robot building is a problem-based learning approach where many different solutions exist for a particular problem [3]. Therefore children tend to improve their creativity skills, ability of logical thinking, understanding of scientific concepts and other social skills like communication, leadership, planning and cooperation apart from technical skills by involving in robot building [4]. As an encouragement robotic challenges and competitions are organized all over the world [5].

The children face many challenges in building robots even they are with the interest to do so. Designing, building and programming even a simple robot from scratch is a difficult task for a kid and therefore many robot kits have been introduced to make robot building and programming easy for children and people who lacks knowledge and practice in electronics, mechanics and conventional programming [3].

Nowadays, there exist several frameworks which support building robots with the use of interactive software and robot kits. Out of the existing construction kits for robot building, the existing robotics kits intended for children are very limited [1]. These kits have reached simplicity, interactivity and extensibility in different levels, but we see that there is more space for improvements in each aspect. Achieving these features within the same solution has become challenging and improving one aspect has led to compromise others.

Most of the existing robot kits require at least basic programming knowledge which is difficult for a child to handle and basic hardware related knowledge which is not known to a child. Complexity in the user interfaces does not create a user friendly environment for child users and some of the GUIs are not attractive and simple enough for them. Since it is not possible to extend the application and the kit for new modules in most of the cases, the robots are restricted to limited functionalities. So, the child will eventually lose interest in doing the same thing without having room for new things. Since most of the available robot kits are commercial products, they are not affordable for most of the children. In order to overcome these problems recognized in the existing solutions, there is a high demand for a simple, interactive and extensible robot kit which aims kids. In this paper we discuss about SiFEB (Simple Interactive Friendly Extendible Bots), an open source robotic development platform and a framework initiated to address these problems, mainly targeting primary and secondary students.

978-1-4799-7910-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ICAIET.2014.32

II. Related Works

A. Building Robots

There are many popular projects and products like Arduino, Dwengo and Raspberry Pi which supports building robots from scratch [3]. To overcome the complexity of building robots from scratch, projects like Lego Mindstorms, Lynxmotion and Vex have been carried out [3]. These robot kits, ease the process of building robots for younger children. The brick concept and snap and friction connections used by Lego has been a reason for its popularity among children. Young children have been successful in building robots by following step by step guides or following tutorials or with the guidance of an adult using Lego like robot kits, but they may face problems in building and programming robots using these kits by their own according to their own interests, due to the characteristics of those systems. Using nuts and screws for the connections of modules in Vex robot kit has made the process of building robots hard for the children. Some other robot frameworks like iPitara[6] face extensibility problems, which will end up with not allowing kids to build robots based on their interests, passions and imaginations. The robot kits like Lego and other recent such projects allow the robot to have customizable capabilities, but the number of capabilities that the robot can have at a time is pre-defined at the basic level. Although there are few solutions which address the problem of extensibility with block modules, it has been difficult to get the appearance of robots as in the imagination of kids using the robot platforms like roBlocks [7] falling in to this category.

A key principle in designing construction kits for children is allowing them to "learn-through-designing", not just by "learn-through-doing" [8]. With "Low Floor and Wide Walls" concept, designers should allow children to work on projects that grow out of their own interests and passions [8]. Some common drawbacks that we can see in existing robot frameworks related to these concepts are, expensiveness of all-in-one kits, difficulty in interfacing with off-the-shelf components, difficulty in finding or buying replacement parts and waste of money in components that are not needed when buying a robot kit as a whole [3]. Also as mentioned earlier, although some robot kits allow to have different capabilities with the use of different types of modules, the number of modules that can be connected to the robot at a time is pre-defined. Therefore most of the time the children would not be able to give a variety of capabilities to the robot as they wish.

B. Programming Robots

Different programming approaches like conventional textual programming, visual programming and tangible programming are being used to program robots. Although tangible programming methods open up more opportunities for children to interactively involve in programming [9], visual programming approaches would be useful and suitable in the context of programming robots while improving the programming concepts in children's minds. Visual programming languages mainly feature simple syntax with graphically nested loops and conditional statements with approaches like allowing children to program by dragging and connecting icons on the computer screen [9]. Different types of visual programming languages like rule based visual programming [10], event driven visual programming, object oriented visual programming, logic based visual programming, finite state machine based visual programming [11], example based visual programming [12] and data flow driven visual programming gives different types of experiences to the user. Consistently high level of enjoyment, ability to postpone the complexity of syntax of textual languages until the student has grasped the fundamentals, instant and vivid feedback are significant benefits of a visual programming language for a child learner [13].

Scratch [14] has become a popular block based visual programming language among children. Limited support for concepts like data structures, procedures are not oversights, but the designers of Scratch have deliberately avoided cluttering the language with anything a child might find threatening [15]. For robot programming many other suitable visual programming languages like LEGO Mindstorm and Modkit for Vex [16] have been developed.

Still there are fairly many researches going on in order to abstract the complexity in programming for children and deciding suitable programming approaches for different children groups of different age ranges and from different backgrounds have become a challenging task.

III. SiFEB Vision and Design Concepts

It is clear that there is a great need and a high demand for a robot framework that can combine simplicity, extensibility and interactivity to make it useful, effective and interesting for the children who build and program robots. Hands on experiences with such a robot framework can increase the interest of small children in the field of robotics as well as in science stream from the very early stages of their childhood. Our idea is to come up with a robotic framework called SiFEB (Simple Interactive Friendly Extendible Bots) to address this problem for the benefit of younger generation.

The important features; Simplicity, Interactivity and Extensibility are considered with respect to main areas:

1) Physical aspects of the robot kit such as modularization, module size and connectivity of the modules.
2) Logical module connectivity related aspects as addressing modules, structuring modules and transferring data among modules.
3) Aspects related to software component such as visual programming languages and providing APIs.

IV. SiFEB Robot Kit

A. Physical Aspects

A main aspect of SiFEB's vision is to design and implement a robot kit for children to learn programming while enjoying robotics, such that children do not need to worry too much about the hardware modules and their details. Therefore we introduce hardware modules to address simplicity problem, which can encapsulate the internal complexity of hardware devices that will be used to build a

robot. The robot can be constructed by attaching modules in such a way that children construct different creations out of building blocks.

Any module can be plugged into another module and each of the added modules will add new capabilities to the robot. This type of hardware architecture can address the extensibility problem recognized in existing robot kits. The number of modules that can be attached together to build a robot will solely depend on the size of the address space that we are using to address the modules and the hardware restrictions like electrical resistance and capacitance. Therefore this design of hardware architecture will be a novel solution to the problem of extensibility, which suffers from predefined number of input and output ports in most of the existing robot kits. But such a design introduces many other challenges like handling communication between modules which we will discuss later in this paper under Logical Module Connectivity Aspects.

These modules are designed in a plug and play manner where the programming interface will automatically detect the newly attached modules and removal of modules. Also we impose the ability of real time debugging of the modules. With auto detection capability and real time debugging capability of the hardware modules in collaboration with the software component can address the problem of interactivity related to hardware aspects. Identifying and learning about the modules and the capabilities of them can be made easier for children with these underlying hardware supports.

We involved in sufficient research in the process of deciding size, shape and connectivity of modules. The initial concept was to have modules in ideal sized cubes or multiples of an ideal size, where each interface of a cube can be used as a connection surface to connect another module. To reduce the complexity of an interface, our interest was in using mechanical connectors which we can use for electrical connectivity at the same time. One such option was using snaps as used in ActiveCube [17]. Other alternative connectors that we paid our attention were spring loaded connectors and magnets. Having two types of modules as male and female for a particular functional module was an easy option, but not a good solution. So we tried to come up with a unique interface with a combination of male and female connectors, which can be plugged into another interface of the same connector configuration. We were able to come up with a successful design of such an interface, where we used a combination of spring loaded connectors and snaps for electrical and mechanical connectivity. Although this proposed design would support building a perfect extensible robot kit, due to possible drawbacks like energy wastage at spring loaded connectors and complexity due to the increased number of interfaces of the modules which require multiples of ideal sized cube, we moved towards a simpler solution. Also the ability to connect modules to any other interface of another module will be a complex concept for a child to handle.

The current approach that we use for our hardware design allows more simplicity compared to the previous concept. A module consists of two interfaces in two opposite surfaces of a module, one interface having female snaps which are used to connect the module on top of another module and the other having male snaps which are used to allow other modules to connect to itself. Each module has a microcontroller in it to facilitate communication and functionality of the device within that module. Since this design of a module restricts orienting a device in the module in a required direction, we introduce another type of module with no intelligence called extenders. They are rotatable and have two interfaces in adjacent surfaces (not in opposite surfaces as in normal modules) to support orienting devices in preferred directions. The Fig. 1 shows the modules that we have created for our first prototype. Our first prototype consists of a main module, a sonar sensor module, a line sensor module and a base module. The main module acts like the brain of the robot and the base modules contains wheels. With the use of those modules, a child can easily build an obstacle avoiding robot as shown in Fig. 2.

B. Logical Module Connectivity Aspects

In order to support communication among modules of the suggested extensible architecture we cannot allocate one port per a module. Since there are limited number of ports in a main controller, connecting a module per a port will allow only to have limited number of modules and each module should be directly connected to the main module which will introduce problems related to space. If one port of the main module can handle many modules and if the modules can be connected on top of any other module, these problems can be overcome. Therefore, in order to facilitate the extensible architecture, we handle communication between each module and the main module using a communication bus. All the modules are connected to the communication bus and the main module which contains the main controller acts as the master which controls the communication within the network of modules. Also the main controller acts as the communicator between the software components and the slave modules as shown in Fig. 3.

In our first prototype we use I2C communication protocol to implement communication among modules using two signal lines for data and clock. Our main controller which acts as the I2C master is an Arduino and we have used Atmel as well as PIC microcontrollers in our slave modules which act as I2C slaves.

Figure 1. Disassembled modules of initial prototype.

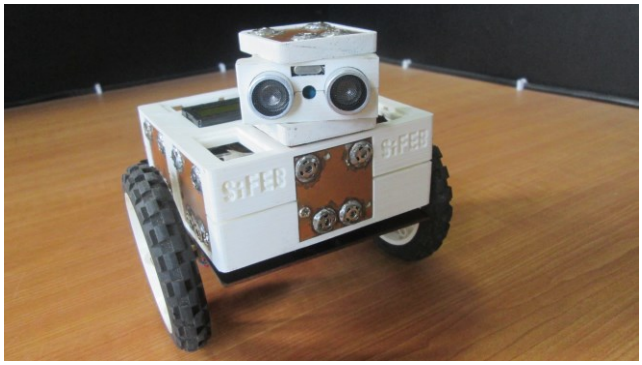

Figure 2. An obstacle avoiding robot using SiFEB's initial prototype.

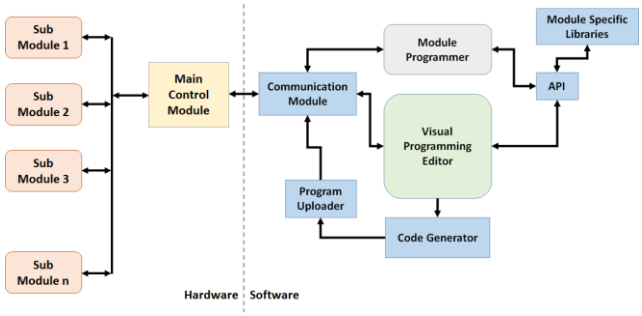

Figure 3. High level architecture of SiFEB robot framework.

To accommodate auto detection of modules at the software component, we should have a way to identify hardware modules. In this case we can use addresses for the modules. Allocating fixed addresses based on the type of the module will help to identify the module and its capabilities, but will not help differentiating two or several modules of the same type that are connected to the robot. Assigning a unique and fixed address per module will solve that problem, but the size of the address space will limit the number of modules that can be manufactured in order to avoid duplication of addresses. Having a fixed address to identify the type of the module and another unique but dynamically assigned address to differentiate a module from others will be a better solution to address this problem. In that way the number of module types and the number of modules that can be combined to build a robot at a time are the only factors that will be constrained by the sizes of the address spaces. There are many alternative mechanisms that can be used to assign addresses dynamically for the modules and we are still in research phase in selecting the best mechanism for our context.

In choosing a mechanism to assign module addresses dynamically, we consider other factors like reducing the number of electrical connections needed, reducing the barriers for extensibility like electrical resistance, rapid address allocation capability and ease of detecting changes to the module network.

Ability to structure the robot with respect to connected modules is another significant capability that can increase the interactivity. With the knowledge on the structure of the robot, the software component can interact with the child user in a more useful and efficient way. This ability can be achieved along with a properly handled addressing mechanism, such that the structure is created or updated at the main controller while assigning addresses or removing addresses according to updates related to physical module connectivity.

C. *Software Component Related Aspects*

The programming interface that we are providing should facilitate very simple, friendly and interactive user interactions that can make robot programming an enjoyable task for the children. The stage concept that we enforce in the programming interface provides different levels suitable for children based on their age and other factors like fluency and technical exposure. From beginner levels through intermediate levels to advanced levels a child can climb a ladder of programming robots from very simple and interactive programming languages to low level conventional programming languages.

The visual programming structure that we have provided for beginner levels is more in to action or task based programming. Instead of using the machine friendly commands, we use commands related to the actions corresponding to the used hardware modules, while interpreting these actions with the use of easily understandable graphics and terms closer to natural language. We use the drag and drop concept and programming blocks concept that is been used in Scratch [14] to minimize the interaction with keyboard which will be an overhead for a child programmer. For example, at the beginner levels instead of allowing the child to control left and right motors separately, we allow giving commands like go forward, reverse, turn left and turn right as abstract commands which hide the underlying complexity. Also this visual programming paradigm allows the children to map capabilities of hardware modules to a context that is more familiar and understandable for them. For example, instead of interpreting the inputs received from a sonar sensor module as detecting obstacle in front of it, it is interpreted as seeing an object in front of the robot. With these features, we help children to visualize the robot as a playmate, which will make them feel interested to involve in programming the robot to complete new tasks. The visual programming interface that we have implemented for our initial prototype is shown in Fig. 4.

Auto detection of plugged and unplugged modules to the robot at the programming interface, encourage children in programming by providing the indications of new capabilities of the robot and by giving suggestions. The friendly personified interface specially designed for the children with interesting visual and audible feedbacks will introduce a novel experience to robot programming. Also these encouraging feedbacks keep the interest of children while they involve in programming and help them to learn and improve their knowledge. The ability to debug actions of the connected modules makes a richer interaction with the child users, because they do not want to complete writing a program to check the abilities that the robot is having.

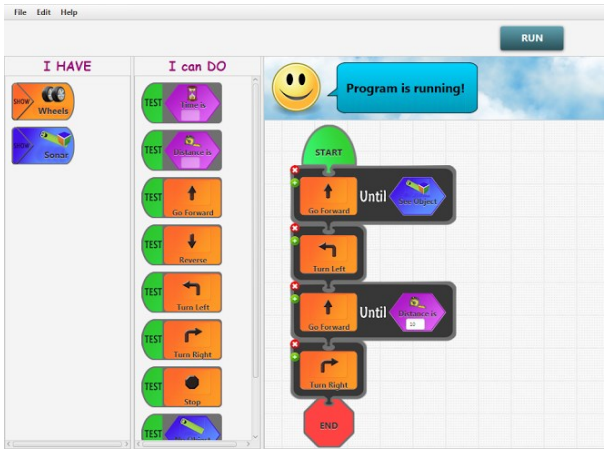

Figure 4. Visual programming interface for beginner level.

The advanced users like developers or experienced child programmers have the ability to modify the capabilities of modules, without just programming with the given set of capabilities. Therefore SiFEB opens opportunities for creative ideas and encourage the users to build robots of their interest without suffering from limitations.

V. Priliminary Results

Initial prototype was used as a proof of concept. We used the prototype's main module, base module (wheels) and the sonar sensor (distance sensor) module to let a group of children aging around 12 years with average computer literacy try it and get their feedback and observations. The software interface too was built to a level that it could be used to create linear programs using the capabilities of the modules. We first demonstrated with explanations a simple obstacle detection program and asked the child to create a program with few steps. After building the robot the child first divided the high level task into single steps and these steps were used to create the program for the robot through our interface. Observations showed that the child could quickly build and program a simple robot using our prototype, but it was also noticed that the child sometimes found it a little difficult to connect modules which implied that connecting interfaces need improvements.

VI. Future Enhancements

Since we wish to carry out SiFEB as an open source project, it will provide the capability to extend the robot kit with new modules and extend the application for new modules. We wish to provide an API for the developers to add newly developed modules to the system. To manage the open source community that will grow around SiFEB we may need to have a proper system which can assign module type addresses for newly developed modules and a resource base where the libraries and other resources related to different types of modules will be maintained. The advanced users can develop new hardware modules with software libraries and add them to the application. Due to such open source community, the kids will always have new things available to try. Also anyone interested can build the hardware parts with less cost and can use the software free of cost.

Although we have several alternatives in implementing dynamic address allocations for modules at run time, we are still involved in research work in order to select the most optimized solution. Also we are still involved in research work related to children's cognition and how we can absorb them the essence of programming and improve their STEM based knowledge through SiFEB. We have thought about possible enhancements like introducing tutorials which appear as different levels of a computer games. Evolving from beginner level of programming approach which is implemented in our initial prototype to advanced level of programming is our focus of next stage of this project.

References

[1] P. Blikstein, "Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future," in Proc. of the 12th International Conf. on Interaction Design and Children, 2013, pp. 173–182.

[2] A. Brooks, J. Gray, G. Hoffman, A. Lockerd, H. Lee and C. Breazeal, "Robot's Play: Interactive Games with Sociable Machines," Computers in Entertainment, Vol. 2, No. 3 , July 2004 .

[3] B. Vanderborght, C. Ciocci, C. Vandevelde and J. Saldien, "Overview of Technologies for Building Robots in the Classroom," in Proc. of the International Conf. on Robotics in Education, Lodz, Poland, 2013, pp.122-130.

[4] S. Bell, "Project-based learning for the 21st century: Skills for the future," The Clearing House, vol. 83, pp. 39-43, 2010.

[5] F. Wyffels K. Bruneel, P. Kindermans, M. D'Haene, P. Woestyn, P. Bertels and B. Schrauwen, "Robot Competitions Trick Students into Learning," in Proc. of 2nd International Conf. on Robotics in Education, Vienna, Austria, 2011, pp. 47-52.

[6] N. Karwall, "Visual Programming Application for Children to program Robotic Toys," in Proc. of International Conf. on 'Designing for Children' with Focus on 'Play + Learn', IDC, IIT Bombay, 2010.

[7] E. Schweikardt and M. D. Gross, "roBlocks: A Robotic Construction Kit for Mathematics and Science Education," in Proc. of the 8th International Conf. on Multimodal Interfaces, New York, USA, 2006, pp. 72 - 75.

[8] B. Silverman and M. Resnick, "Some Reflections on Designing Construction Kits for Kids," in Proc. of the Conf. on Interaction Design and Children, Boulder, Colorado, 2005, pp. 117-122.

[9] T. Sapounidis, and S. Demetriadis, "Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences," Personal and Ubiquitous Computing, vol. 17, no. 8, pp. 1775-1786, Dec. 2013

[10] M. B. MacLaurin, "The design of kodu: a tiny visual programming language for children on the Xbox 360," in Proc. of 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, Austin, 2011, pp.241-246.

[11] J. Mattila and A. Väätänen, "UbiPlay: an interactive playground and visual programming tools for children," in Proc. of 2006 Conf. on Interaction design and children, Tampere, Finland, 2006, pp.129-136.

[12] N. Guibert, P. Girard and L. Guittet, "Example-based programming: a pertinent visual approach for learning to program," in the working Conf. on Advanced Visual Interfaces, Gallipoli, Italy, 2004, pp.358-361.

[13] A. Wilson and D. C. Moffat, "Evaluating Scratch to Introduce Younger Schoolchildren to Programming," presented at 22nd Annual Workshop of the Psychology of Programming Interest Group, Leganés, Spain, September 19-21, 2010.

[14] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," Communications of the ACM, pp. 60-67, November 2009.

[15] B. Harvey and J. Monig, "Bringing 'No Ceiling' to Scratch: Can One Language Serve Kids and Computer Scientists?," in Proc. of Constructionism 2010, Paris, France, 2010.

[16] A. Millner and E. Baafi, "Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects," in Proc. of the 10th International Conf. on Interaction Design and Children, Ann Arbor, USA, 2011, pp. 2–5.

[17] R. Watanabe, Y. Itoh, M. Asai, Y. Kitamura, F. Kishino, and H. Kikuchi. 2004. "The Soul of ActiveCube: Implementing a Flexible, Multimodal, Three-dimensional Spatial Tangible Interface," ACM Computers in Entertainment, vol. 2, no. 4, Oct. 2004.