UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER

COMPUTER SCIENCE HONOURS
FINAL PAPER
2016

Title: Educational Robots as Pedagogical Tools in Computer Science Extended Degree Introductory Course

Author: Joshua di Bona

Project Abbreviation: RaspiEd

Supervisor(s): Gary Stewart

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	10
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	15
System Development and Implementation	0	15	15
Results, Findings and Conclusion	10	20	10
Aim Formulation and Background Work	10	15	10
Quality of Paper Writing and Presentation	10		
Quality of Deliverables	10		
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	
Total marks		**80**	

Educational Robots as Pedagogical Tools in Computer Science Extended Degree Introductory Course

Josh di Bona
University of Cape Town
dbnjos001@myuct.ac.za

ABSTRACT

There has been a significant increase in the interest and use of robotics in education over the last sixteen years, especially where robots are used as pedagogical tools to improve the learning experience [4][5]. This research paper looks at creating an educational robot system which could be implemented into the University of Cape Town's CS1 extended degree course to help increase the motivation of the students when using the tool to learn the basic concepts of programming. If this is effective this could potentially see better results as well as drawing more people into CS majors. Previous studies have shown a decline in the number of students choosing CS as a major [8] and this could be the first step in preventing this from happening while trying to retain the interest of the students who have had to decant to the extended degree course. The system is designed to be as similar to the current tool used in the course, Python Turtle [9], while still being an independent tool that could be used in any CS1 course. The educational robot platform created allows students to program a robot through a web-app whilst being able to watch the code run remotely via a live stream. The robot is controlled by a Raspberry Pi and fitted with an array of sensors. The system was tested with the help of seven expert users, across three weeks with two rounds of testing. Participants were asked to complete a set of tasks relating to a specific CS concept, containing four questions each with one of the robot's sensors as its main focus. Usability tests were done and the motivation of the robot measured and compared to other simple tools. The questionnaire used is designed by McGill and measures motivation through four components consisting of Attention, Relevance, Confidence and Satisfaction [1]. The analysis of the results showed an increase in Relevance and Confidence and a significant increase in the Attention and Satisfaction of the participants when using the robot. The results are satisfactory and give cause to test the system further and possibly introduce it into the curriculum in the near future.

CCS Concepts

•Social and professional topics→CS1 • Computer systems organization→Robotics

Keywords

Educational Robot; Computer Science, Programming; Pedagogical tools; CS1; Undergraduate; Extended Degree; Python; UCT.

Month 4–10, 2016, University of Cape Town (UCT), South Africa

1. INTRODUCTION

Over the last few years, robots have not only become a new topic for research and study, but have found use as a modern pedagogical tool [4]. This is seen in the sciences with engineering students learning about the hardware and computer science students using them to learn to program. The use of robots as a pedagogical tool has seen many responses, mostly positive, suggesting that there was an apparent effect of an increase in interest in the concepts being taught [3], however, there do exist situations where their use wasn't proven to be beneficial or the results were rejected as no validated and widely accepted methods were used to test for any effects that the use of robots had on the students, whether it be on learning or simply their attitudes towards the material when being taught with robots. The use of robots as a pedagogical tool supports the idea of learning-in-context [4], where it has been proven to be more effective to teach students, especially the basic concepts in a subject, [6] by using a real life example as an educational tool. Computer Science is also fundamentally about problem solving, and the examples shown to students in class or given to students as assignments can profoundly influence how they perceive CS relevance and use-fullness. Therefore, allowing the use of robots to attract those students who had previously been intimidated, excluded or weeded-out of the field [7].

We have seen the use of robots' increase over the years, mainly in the Engineering Department, however not in the Computer Science Department. Given the expense and work required to introduce the use of robots into a new curriculum for CS1, there exists precedent for further testing and comparisons to Python Turtle [9], the current educational tool being used. This paper proposes that robots could be beneficial to learning the basic concepts of computer science, by increasing student's motivation to learn, especially in the CSC1010H course, which is an extended degree program of the mainstream CS1 course. Research was done into a few of the previous experiments that had been done in the area in order to see what worked and what didn't when introducing robots into the curriculum. In the end it was proposed that a complete Educational Robot Platform would be built, consisting of a robot, built 'from scratch', by combining different inexpensive parts, which has two wheels and offers a range of movement functions, similar to those provided by the student's current tool used in lectures, Python Turtle [9]. In addition to the movement features, the robot will also make use of a number of different features, including an ultrasonic sensor, a camera and an infrared sensor to make coding on it more entertaining. Finally, the robot, along with its features, will be made available to students through a supporting web app, which will allow students to access the robot from off campus, allowing them to write code,

submit it and view the robot's movement through a live video stream.

The project itself consists of three parts and three papers by the project members, which relates to the three separate parts making up the Robot Platform, these include the robot itself and the movement and control of the robot, the sensors and additional features and finally, the web-app. This paper focuses on the first part, relating to the movement and control of the robot while trying to answer the research question proposed. The paper on the features and sensors is written by Jeremy Coupland who is testing which sensors and features pair up best with the different CS concepts. The paper on the web-app is written by Muhummad Patel who is testing whether the web-app is easy to control the robot from and able to allow one robot to work with a whole class of students. I am responsible for the building of the robot and its enclosure, the creation of the movement module, the wrapper class as well as half of the wiki.

1.1 Research Question

The research question proposed in this paper is "Can an educational robot, along with a high-level programming interface, be used in the UCT CS1 extended degree program to increase the motivation of the students learning basic Computer Science concepts?". Along with usability testing, testing is done with expert users in order to measure the effects of the robot platform on motivation and to provide a professional opinion and insight into whether further testing or implementation would be something that could be viable in the future. The evaluation of motivation is done using a validated motivation survey developed by McGill [1] and the usability evaluations done using the validated System Usability Scale (SUS).

1.2 The Raspied Educational Robot Platform

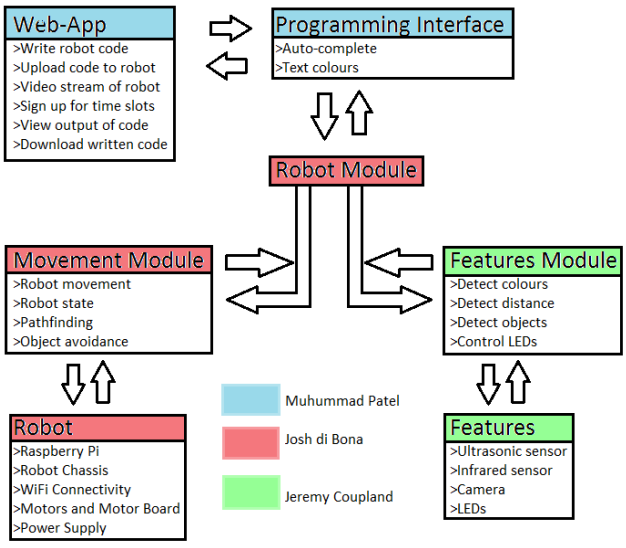

Figure 1: Educational Robot Platform System Diagram, depicting the three separate parts of the project and how they link.

The Educational Platform built for this project and discussed in this paper consists of a 2-wheeled dc motor robot, controlled by a Raspberry Pi. In addition, it has a camera, a set of LED's, an infrared sensor and an ultrasonic sensor. The robot is headless and can be controlled wirelessly from a web-app, which controls access to the robot through the use of time slots and allows code to be run on the robot and even typed up in the web-app itself. The web-app shows the output of the code as well as a live stream of the robot as it runs the code.

1.3 Project Significance

If we can get a better understanding of if and how robots might help in the extended degree program and increase motivation it may eventually be a feasible plan to introduce them to the mainstream CS1 course with a bit more testing. With the sudden drops in cs1 majors in other places in the world [8], it may prove worthwhile to implement measures to help prevent this before it happens here as well as decrease the number of students needing to move to the extended degree program.

1.4 Project Aims

The CSC1010H course is aimed at students struggling to understand basic concepts on a deeper level [8]. As well as learning to program, students should gain from the course a solid understanding of the scope of computing and the role of programming [10]. There is a lack of a robot with accompanying software that caters for struggling students specifically. Students bring with them a wide spectrum of backgrounds [11] and this causes CS1 general courses to contain a broad mixture of students with varying abilities. All of the current research, and those analysed in the literature review, have been done on CS1 or similar populations. There is a lack of testing to see the effects of using robots to teach basic concepts to those struggling to understand them and it gives the opportunity to be able to see the true benefits in a focused test group and its potential benefits to a wider audience when measured with validated questionnaires. The end aim of the research is to be able to have an understanding and to be able to make a professional call as to whether robots are something that the Computer Science Department should look into trying to fit into their curriculum in the future.

2. BACKGROUND

2.1 Educational Robots

The type of robot referred to in this paper is a type of physical electrical mechanical agent that can be programmed to complete a series of tasks. Robots have been around for a while but are still seen as futuristic technology, however, educational robots have made a rise in the last decade with programmable robots being made for all different ages and available from major stores.

There are a few different meanings for the word 'Robot', this paper talks about the type of robots used in teaching and learning. These are known as pedagogical tools, which is what this project is attempting to build.

There already exist a few robots like this, for example there exists the e-puck robot which is designed for education in engineering [14]. Another example is the Lego Brick Sculpture which is a controllable robotic arm [15]. The Scribbler is an example of a robot similar to the one designed in this project that was designed to be inexpensive, portable and robust. There are quite a few features available on the robot, such as multiple motors, proximity sensors, line sensors, programmable LEDs, a color camera and a pen port. It also works with the high-level programming language Myro [12] [13].

These robots made for education in computer science usually have simple hardware that can perform fairly simple actions allowing the hardware to be simple but have the freedom for tasks to be as challenging as you like. Other than teaching basic concepts, it was found that robots also come in handy when teaching more complex concepts such as AI and multi-agent learning [7] [16]. In

most cases the robots and the frameworks, if any, are built in such ways that they can be used for learning more complex concepts, or even come with methods in the framework to help simplify the process [17]. These modules enable and encourage the exploration of more complex robot control paradigms such as robot learning and robot vision [16].

2.2 Robots in Computer Science Education

Fagin's research has been one of the main inspirations for this project [18]. His research looks into the idea of teaching computer science using robots in CS1, specifically Lego Mindstorms, as well as looking into how one goes about measuring the effectiveness of the robots in teaching computer science. His paper looks at the experiences of using a high level language and robots to teach some basic CS concepts and how they might be employed to teach a particular idea. He looks into the teaching of sequential flow, variables and constants. Teaching a class of 180 students using the robots, while another 800 students follow the original course. He finds that robots are useful when teaching procedures, as students quickly learn that a few moves of the robot can be saved for quick use later on, such as having the robot turn right.

He concludes that robots provide a very important 'hands on" learning experience and the design-write-run-redesign feedback loop is very fast, which students really enjoyed. Having taught the class both with robots and without, Fagin does comment that when taught with robots, the fun-factor is much higher. The analysis of the results of the two classes showed worse results for the students who were taught using the robots, the reason given was that the students weren't able to access the robots from home and were thus being deprived of the opportunity to work on their code and practice the write-run-debug loop, which has proven to be an important element to learning. For future work he suggests more testing be done in order to attempt to separate the effects of the robots on learning from the effects of reduced access to the robot [2][3].

2.3 Robots and their Effect on Motivation

McGill focuses on this topic and in her study of related work, consisting of other research papers which attempted to analyze the effects of robots on motivation, puts together a validated design and study to properly analyze motivation. She suggests that motivation is made up of four elements consisting of Attention, Relevance, Confidence, and Satisfaction.

The problem McGill is trying to solve is the lack of motivation in the students and the deceasing lack of interest in programming and the number of students specializing in development. In her study McGill attempts to answer four questions: "How does the use of robots throughout the course affect student motivation?", "Is there a relationship between motivation levels of students learning programming with the robots and gender?", "Is there a relationship between motivation levels of students learning programming with the robots and their technical self-perception?" and "Is there a relationship between motivation levels of students learning programming with the robots and their interest in development?" [1]. This saw the introduction of a new curriculum and layout to the existing course, which uses robots to introduce the students to programming. The IPRE robot (Scribbler and Fluke) was chosen as the tool for teaching programming, which was created as a hands-on tool for introducing students to programming using the Python scripting language and the Myro package. The research questions were explored using descriptive research methods in a pre-experimental design. A pre and post-survey were disseminated to determine the participant's motivation using robots in the introductory programming course. The pre-survey attempted to judge a participant's interest in robotics and attitude towards learning to program as well as measure motivation.

McGill found that the use of robots had a positive effect and influenced the student's attitudes towards learning but found it had little or no effect on relevance, confidence or satisfaction. Fagin found negative results in his study which is believed to be because of the lack of a sufficient code-test-debug loop, as the students couldn't test the robots at home or in their free time. However, both found that the idea of programming with robots captured the attention of the students and this in turn increased their motivation in the classroom [1] [2]. The experiment had a few threats to validity, among others are the lack of a control group to prevent the effects of learning. Additionally, the fact that the pre-course survey may have biased participant's views of the course should be considered as it may have planted the idea that robots are interesting and even make students more curious.

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1 Requirements Gathering

3.1.1 Communication with Supervisor

There was a constant channel for communication open with the supervisor. Meetings were held once a week at minimum where the current state of the project was discussed and where ideas were brought to the table which in some cases saw old features rejected and new requirements added to the software or the hardware. Because the project is working with software and hardware, new requirements were often thought of as the project progressed.

3.1.2 Comparison with Current Software

The students in the extended degree program currently use Python Turtle as a learning tool. They complete assignments using it and are able to use it in their free time. It contains a GUI interface allowing students to control a 2D turtle around a screen with simple high-level methods such as move() and turn() [9]. Because this project uses an experimental design where the students who are going to test the robot would have already used Turtle, it was decided that the robot be as similar as possible when it came to the software. Additionally, it was taken into consideration that it might eventually be gradually introduced into the curriculum if necessary, with the code being run on Turtle and thus not having to be changed much for it to be able to run on the robot. Therefore, the features of Python Turtle and what it offers to its users formed some of the basic requirements of the software for the movement of the robot.

3.1.3 Hardware Requirements

The hardware was required to be cheap and cost-effective as well as fitting our requirements of what it could do and what features it had, so instead of buying a complete robot such as the expensive Lego Mindstorms [18], a robot was put together from scratch, by combining different bought parts. These requirements for the hardware were set before the project began. The robot had to be able to work without any supervision since our platform allowed wireless control through a web-app and the robot had to be headless, meaning it had to be able to be controlled wirelessly and have its own power supply.

3.2 Development Framework and Methodology

3.2.1 Programming Language and Framework

Python was the language of choice as the robot was built using a Raspberry Pi, which is preloaded with Python and is its official programming language, as well as the language taught in the extended degree program. A Module or Application User Interface was built, which is imported by a wrapper class and then in turn by the students, allowing them quick and easy access to all the provided features. Just like they had to do with Python Turtle before.

3.3 Hardware Development Methodology

3.3.1 Iterative Development

An iterative approach was taken towards the development of the hardware, this allowed for team members to regularly touch-base and ensure that all components are compatible. All three sections of the project required the use of hardware for testing purposes and each section had to be compatible with each other. At first all three sections were developed on separate Raspberry Pi's but as the project progressed and each section had to come together, testing had to happen each time a new component was added or something changed. Because of the uncertainties of hardware, everything had to be retested and the compatibility checked. The robot was first developed with just its motors for movement and later on the sensors added one by one and then the robot with its features was made compatible with the web-app.

3.3.2 Challenges

Working with the hardware was the biggest challenge. There is no hardware present in the teaching of the undergrad computer science degree and is more present in the engineering degrees. So having to learn how the hardware worked, learning how to solder and read up on Raspberry Pi and its pins proved to be a hurdle at the beginning of the project.

Getting the motors to go straight was a major issue in the development of the robot and the movement module. Even when the software is theoretically working fine, the hardware might act up and trick you into thinking something was wrong with your code. The hardware never did the same thing twice which made testing challenging, when one error was corrected another was discovered. Another small challenge was testing the robot. Because the software could only be tested wirelessly on the robot, the code-test-debug cycle of development was quite slow which the addition of the web-app at the end would hopefully speed up. The robot had to be headless for the project's needs so wireless control with low or no maintenance was needed therefore, research during development was done on how this could be achieved which involved the testing of different options.

3.3.3 Approach

The process used to meet the requirements during the building of the robot came mostly from trial and error; when one solution didn't work another was found. Because we aren't Engineers, a lot of the hardware proved to be rather technical and required some consulting from the UCT engineering lab as well as online resources when tutorials that were meant to work didn't and when soldered boards had no response.

Because the hardware didn't always behave like it was supposed to, even when the software controlling it was theoretically correct, quite a bit of testing and experimenting had to be done to get the motors running at the perfect speed or the robot turning the correct number of degrees.

The finished robot finally consisted of the following parts: A Magician Chassis Kit - consisting of two acrylic chassis plates, two 6V DC motors and two 65mm wheels. Additionally, a Raspberry Pi 3B and a RaspiRobot Board for the control of the motors were used. Power was provided by a Romoss Sailing 4 10400mAh power bank with two 5V USB connections, a 2.1A to power the Pi and a 1A to power the RaspiRobot Board. This power bank was chosen as it was the biggest storage that could fit on the robot and wasn't too heavy. It can also power the robot for a full day of use. The Robot's features include two small breadboards, a camera, an infrared sensor, an ultrasonic sensor and three coloured LED's.

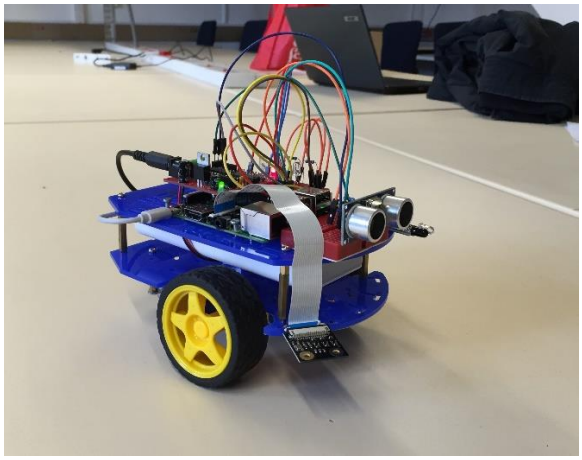

Figure 2: Raspied Robot

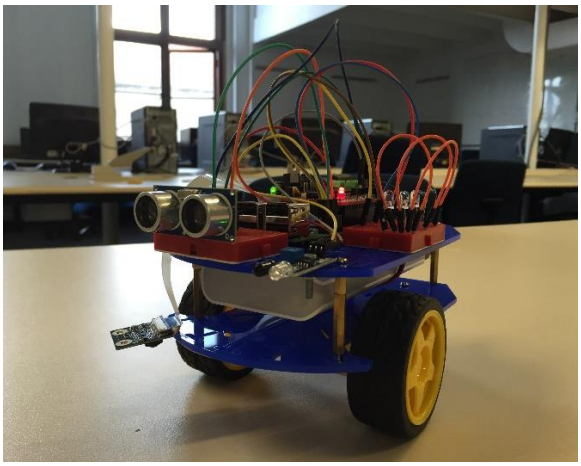

Figure 3: Raspied Robot

3.4 Software Development Methodology

3.4.1 Agile Development Methodology

There was a need to constantly be able to change scope due to uncertainties of hardware. After each iteration, physical testing with the hardware had to be done, which usually uncovered a few more flaws in the software which had to be resolved or the hardware didn't work as expected.

3.4.2 Iterative Development and User Centered Design

Because the software being developed is for students to use, it is best to develop it around their needs. It's no use developing software which works great but is unpleasant to use, as this would directly affect the student's motivations when using the robot.

3.4.2.1 Usability Evaluations

Because the system is being designed for other people, specifically students to use, in order to improve their learning experience, it was important to test the usability of the system. In order to do this the feedback received during the testing process was noted and the participants were given a System Usability Scale (SUS) questionnaire so that the system could be scored, its weak areas detected and improvements be made so the system could be retested using the same methods.

3.4.2.2 Testing, Documentation and Maintainability

Because the software is for students and for use in a course, proper documentation was needed to accompany the code, consisting of a code wiki along with example code blocks. To help with the maintainability of the code itself the code was properly commented, and programmed in an object orientated manner. A text file accompanied the code with instructions on how to simply change the size of the grid world the robot used as well as its speed and other variables which affect its movement and control.

3.5 System Design

3.5.1 Requirements

The following requirements for the movement and control of the robot were decided on as the project progressed.

The robot had to return to the start position with the same rotation after a user finished running code on it. This was done to enable the same code to be run twice and act in the same way, without intervention from the user. The software had to keep track of the position of the robot, as well as the rotation at all times. It had to allow the user as much freedom as possible, while being as similar to Turtle as possible. It had to make the methods as simple as possible with the use of a high-level abstracted module, while managing everything else behind the scenes. Additionally, it had to keep track of the position of the robot in case of the event of a forced stop as well as to allow the position to reset without user's having to manually code it in. The robot was also required to introduce students to some higher level concepts as well as avoiding obstacles when returning to the start position, creating methods to help simplify mundane tasks, keeping the robot within an enclosure and not allowing more than one user to use the robot at a time. Finally, it was also required to protect the hardware from damage.

3.5.2 Approach

The robot module was to be made as similar to Turtle as possible so the movement commands of the robot were kept the same and made to act in the same way. Methods were made to move the robot forwards and backwards - each movement moves the robot one step or block forward. Two methods were made to rotate the robot, either left or right a number of steps, each step being 45˚.

The speed of the robot was set to a moderate speed to still allow the run process to not be too slow, but not so fast that damage could be caused to the robot, or cause the batteries to discharge too fast. It was also important that it move at a speed which allows the robots actions to be easily viewed from the live stream in the web-app. To help fulfill that requirement, a small pause was added between all movements, so for example if the robot is told to move 3 places forward, it moves three steps, with a pause between each step so that its movement can be better visualized.

A major part of the movement module was to constantly keep track of the robot at all times. In order to help do this the software was built to work with a grid-based system. Each movement the robot performed would move the robot into the next block, allowing the robot to always have a grid position. In addition, the software allows 45˚ turns, each turn will face the robot towards another block. Figure 4 below demonstrates how the robot would be able to move in eight directions. Its rotation is then constantly stored and updated, along with its position, as a variable between 0 and 7. This allows us to constantly tell where the robot is and where it is facing at all times, allowing it to be easily sent back to its start position.

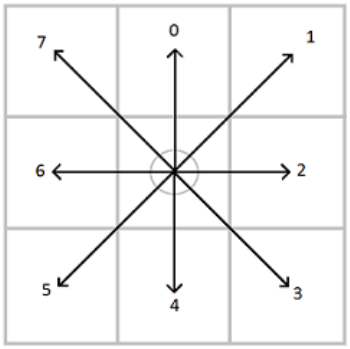

Figure 4: Diagram showing the possible movements of the robot as well as the number assigned to each specific rotation.

In order to prevent the robot from being able to move out of bounds, the robot is given a set grid size, which can be easily adjusted, which it permits movement inside to act as a virtual enclosure for the robot. Each movement called on the robot will check its future position before it moves to check for any out of bounds issues. The grid starts at position [0,0] at the bottom left as in Figure 5 below.

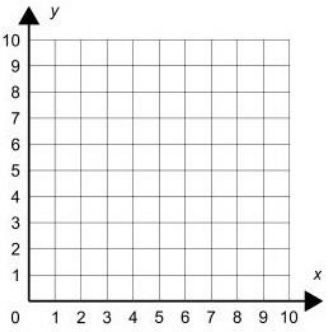

Figure 5: Example of robot grid.

A wrapper class was made to combine the movement module and the features module into a complete robot module, which hides private features from the users and allows for quick and easy changes to be made to the user facing methods. In order to prevent users from making more than one instance of the robot module, the wrapper class was made into a singleton class, so that every instance of the robot made, will have the same co-ordinate position and rotation. A* Pathfinding was implemented to allow

the robot to return home whilst avoiding obstacles as well as providing a quick and easy method for the robot to be moved from one place to another. A method was also made that easily changes the rotation of the robot from its current rotation to a desired rotation position, rotating the robot in the direction that's the shortest. In terms of making the robot headless, it constantly updates its assigned IP to a GitHub Gist file, which can be easily seen by the web-app for access to the robot via ssh.

3.5.3 Features

Since this paper evaluates the entire project, the educational robot platform, this section will mention all the features of the platform in general and not only those of the movement and control section.

The web-app was made to link students to the robot and allow them to control and program the robot from a remote location. It works on a time slot basis and allows admins to whitelist students, who can then register and sign up for hour long time slots throughout the day to a limit. The web-app allows students to open scripts in a built-in text editor or to code right there, and once they are satisfied, run their code on the robot. When they run their code they are shown a live stream of the robot as well as all the output from running their code including any errors that the robot returned. There is a save option to save any scripts you create as well as a re-run button to quickly re-run your code and finally a stop script button for emergencies when the user accidently creates an infinite loop, which will stop the robot, and reset it from where it stopped to its set start position.

The features of the robot include an ultrasonic sensor which can measure the distance to the nearest objects: a set of three LED's of different colours, red, blue and yellow; An infrared sensor which can detect whether an object is immediately in front of the robot, and finally a camera which is underneath the robot allowing the capturing of images. The features module makes a set of methods that make use of these sensors available to the user. These include detecting if there is an object in front of the robot, detecting the distance in cm to the nearest object, turning each LED on or off, and flashing them and using the camera to detect the colour of the grid block beneath the robot.

The movement module, besides managing the position and rotation and bounds of the robot behind the scenes as well as returning the robot back to the start, also makes a set of methods available to the user. These include the following: forward(x) moves the robot x steps/ or blocks forward, reverse(x) moves the robot backwards x blocks, right(x) and left(x) rotate the robot x steps in its position. There are also methods which return the position or rotation of the robot to the user as well as a method to change its rotation to a desired position. The user is able to add obstacles to the virtual world of the robot, and print out the grid showing these. Pathfinding methods are offered which allow the user to enter a start and end destination and be returned a set of nodes which contain the position of blocks to move to. A follow_path() method was made to then allow students to follow the returned path. A third pathfinding method was made which simply takes in a destination co-ordinate and moves the robot to it. All the pathfinding methods avoid obstacles which the student has added to the virtual world.

A wiki is made available which contains all the methods made available in the wrapper class. These include all the movement and feature methods mentioned above. There is a table of methods, showing input and outputs of each method along with individual sections, which give a more in depth description as well as example code blocks. Figure 6 below shows the robot in its enclosure, with the visible grid and the coloured grid blocks.

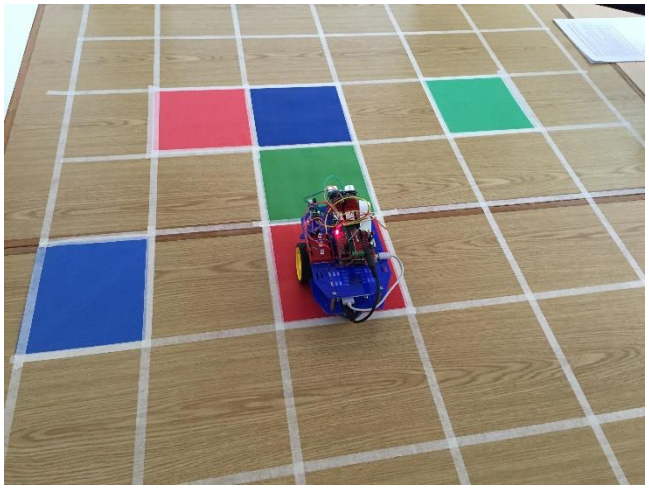

Figure 6: Raspied Robot on the marked out experiment area.

4. SOFTWARE USABILITY

Software usability testing was done in order to test the usability of the system, which is designed for use by first year students. The survey was given to participants after each round of testing. The system was scored each round and general comments recorded, and the changes were implemented before the second round. The results for the usability tests are below.

4.1 System Usability Scale

The Systems Usability Scale was used, which consists of ten Likert scale questions with five response items from Strongly agree to Strongly disagree, measuring the usability of the system in question.

Both rounds of testing were awarded an above average score. The scores were calculated for each survey and an average score calculated for each round of testing. The score after the first round was 87.857% and the score after the second round was 87.917%. So we see a small increase in usability scores, but basically the same, which is good since nothing much had to be changed in terms of the usability of the movement and control section. Two method names were changed, a method removed, a new method added and the wiki made more user friendly.

4.2 General Feedback

In addition to the ten questions, there was space for any additional comments or suggestions for changes to be made. During testing one member of the project group was also assigned to write down all comments from the participant while they completed the set of tasks. Some of the comments given during the first round were, "It is easy to control the robot, and it moved as expected." And "The documentation was very detailed and it was easy to see what most functions did." and that one of the method names needed a better explanation of what it did. The second round of testing saw comments such as, "It is easy to control the robot and the documentation is clear."

5. EXPERIMENT DESIGN AND EXECUTION

In order to test the robot and to be able to statistically analyze the results and to be able to make future work suggestions, a set of tasks were developed to test the robot platform against the basic CS concepts taught in first year.

5.1 Overview of Experiment

The experiment had a group of participants partake, one at a time, in an hour long slot of testing with the robot. The participant was asked to complete a set of assignments relating to a chosen CS concept. Each concept which consisted of either Conditionals, While Loops, For Loops and Functions each had a task for each of the 4 sensors on the robot, the Camera, LED's, Infrared and Ultrasonic. Before testing is started, each participant completed a set of forms, consisting of an ethics clearance form, and a pre-questionnaire testing motivation of the tool they used to originally learn how to program, or the tool they used to teach the basic concepts of CS to their students. After the assignments were completed, which were done on a laptop using the web-app in the presence of the robot and its enclosure, the participant was given a post-questionnaire testing the motivation of using the robot to learn the basic concepts of CS, as well as three System Usability Scale (SUS) surveys testing the three separate sections of the project, and a survey about the features section. This routine of testing was conducted twice, week one had the seven participants complete their first round of testing, during week two, the feedback given in the usability surveys and during the testing was implemented, and the second round of testing ran in week three. Participants were given a different set of assignments relating to a new CS concept each round.

Testing took place in the Honours lab at UCT in the Computer Science building. See Figure 7 below for the set-up used.

Figure 7: Image showing the experimental setup.

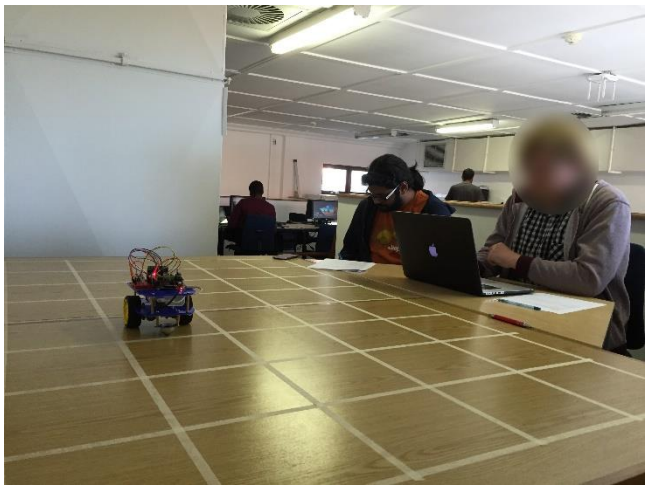

Figure 8: Image showing an experiment in progress.

This paper focuses solely on the usability of the movement of the robot and the pre and post motivation questionnaires. See the other two related papers for information on the results of the features and the feedback for the web-app.

5.2 Participants

The original plan was to have students from the CSC1010H extended degree course test with the robot. They would have all already used Python Turtle so a direct comparison of motivation of their current tool and the robot could have been made. There are benefits of using a controlled group as opposed to the CS1 group as the CS1 group has a broad variety of skills and knowledge. It would also be more feasible to implement and test through a year of the extended degree course as it's a smaller class.

However, due to external factors, the students in CSC1010H as well as those in the CS1 class were unavailable for testing, and alternate arrangements had to be made on short notice. We decided that testing with expert users would be the next best thing. We recruited two CS Masters students as well as five CS Honours students who had previous experience as a tutor in the department. This decision was made as these expert users would be able to put themselves in the shoes of their students or themselves when they first learned the basic concepts of programming and are able to give professional feedback and answer the questionnaires using their experience of teaching or tutoring to impersonate a first year student.

5.3 Pre and Post Questionnaire

It was originally decided to use a post questionnaire on the students after they did testing and to use another on students who weren't tested. But because of the small size of the course, and the testing happening towards the end of the year as well as it not being compulsory, it was decided that it would easier to give students a pre and post questionnaire before and after they complete the experiment and to take into account the effects of a pre and post questionnaire. The same plan was conducted with the expert users, who too, were limited in numbers. The pre and post questionnaire used to measure motivation of the participants is a validated questionnaire created by McGill. The basis of the questionnaire was the Instructional Materials Motivation Survey, which was chosen as the instrument as it was designed to measure whether a media device or tool introduced during instruction for the specific intent of motivating students actually succeeds in doing so. [1]. The questionnaire is broken up into four separate

components: Attention, Relevance, Confidence and Satisfaction (ARCS). The questionnaire has 22 questions in total, with 8 relating to Attention, 5 to Relevance, 7 to Confidence and the last 2 to Satisfaction. Each question is answered by the participant with a Likert Scale on the range of (1-Strongly Disagree, 2-Disagree, 3-Neutral, 4-Agree, 5-Strongly Agree). The original pre-questionnaire which asked participants about their motivation when they use Turtle had to be changed to work with the new expert users. So before the users were given the questionnaire they were given a short note to read when stated, "As an expert user, put yourself in the shoes of your students, or yourself when you first learned the basic concepts of programming. Let 'X' be a tool you have used to teach these basic concepts or a tool you used yourself at one point and answer the questions as best as you can by using your experience of student's reactions to their tools or your own recollections and/or expert opinions". The pre questionnaire was the same as the post, the only difference between the questions was that the pre used 'X' as the subject in the questions while the post referred to the robot.

5.4 Task Design

When designing the set of tasks which each participant would complete, the goals were to create tasks for each CS concept with each of the sensors as the main focus most of them using the movement of the robot in the background. These were as similar to the tasks that any first year in the extended degree program could be given to do when being taught these concepts [6]. The tasks were kept short to allow the user to experience more of the system in such a short amount of time.

5.4.1 Conditionals

5.4.1.1 Infrared Sensor

Write a program that will move the robot forward 3 blocks and then use the Infrared sensor to check for an obstacle. If an obstacle is detected, turn right 2 steps and move forward 1 block otherwise, reverse 1 block.

5.4.1.2 LED's

Write a program that will move the robot to position (4,5) in the grid, and then generate a random number in the range [1,3]. If the number is 1, move the robot forward 1 block and flash the red LED light. If the number is 2, leave the robot where it is, and flash the blue LED light. If the number is 3, make the robot reverse 1 block and flash the yellow LED light.

5.4.1.3 Ultrasonic Sensor

Write a program that will use the Ultrasonic sensor to detect the distance to the nearest object. If the object is further than 25cm, move forward 1 block. If the object is closer than 25cm, turn right 2 steps and move forward 1 block.

5.4.1.4 Camera

Write a program that will pathfind to the following positions: [0,3], [1,5], and [4,5]. At each of these positions, use the camera to take a photo of the grid block and then detect the colour of the photo. If the photo is red, print "This is red!" to the terminal, if the photo is green, print "This is green!" and if the photo is blue, print "This is blue!".

Pathfinding hint: follow_path(pathfind([get_pos()], [goal x, goal y]))

5.4.2 For Loops

5.4.2.1 Infrared Sensor

Write a program that will loop 4 times. At each iteration:

- use the Infrared sensor to check for an obstacle
- If you detect an obstacle, reverse 1 block
- Otherwise, if there is no obstacle, move forward 1 block
- Finally, wait for one second before continuing on to the next iteration.

Once the loop has finished, if an obstacle has been detected at any time, reverse 1 block, and if not, move forward one block.

Hint: Use time.sleep(1) to insert a 1 second wait (don't forget to 'import time')

5.4.2.2 LED's

Write a program that will loop 4 times. At each iteration, move the robot forward, turn right 2 steps, and flash the LEDs in the order red -> blue -> yellow.

5.4.2.3 Ultrasonic Sensor

Write a program that will loop 5 times. At each iteration use the Ultrasonic sensor to calculate the distance to the nearest object. Once all measurements have collected, calculate the average. If the average is greater than 30cm, move forward one block, otherwise reverse one block.

5.4.2.4 Camera

Write a program that will loop 3 times. At each iteration, use the camera to take a photo and then detect the colour of the photo. Print the colour to the terminal, and then move the robot forward one block.

5.4.3 While Loops

5.4.3.1 Infrared Sensor

Write a program that will use a while loop. At each iteration, move the robot 1 block forward and use the Infrared sensor to detect obstacles. If an obstacle is detected, turn the robot 2 steps to the left. Once 2 obstacles have been detected, terminate the loop.

5.4.3.2 LED's

Write a program that will use a while loop. At each iteration, generate a random number in the range [1,3]. If the number is 1, flash the red LED, if the number is 2, flash the blue LED, if the number is 3, flash the yellow LED. Terminate the loop once the red LED has been flashed 5 times.

5.4.3.3 Ultrasonic Sensor

Write a program that will use a while loop. At each iteration, move the robot 1 block forward and use the Ultrasonic sensor to calculate the distance to the nearest obstacle. If the obstacle is closer than 30cm, terminate the loop.

5.4.3.4 Camera

Write a program that will use a while loop. At each iteration, move the robot 1 block forward, use the camera to take a photo, and then detect the colour of the photo, printing the colour to the terminal each time. Once a red block has been detected, terminate the loop.

5.4.4 Functions

5.4.4.1 Infrared Sensor

Write a function checkSafe(x) which will take an integer x as a parameter. The function must use the Infrared sensor to check for obstacles, wait for x seconds, and then check again. If an obstacle was detected both times, make the robot reverse one block, otherwise move forward one block.

5.4.4.2 *LED's*

Write a function lightShow() that will turn the LEDs on and off in a specific order. The order is entirely up to you, just try to keep your function under 25 lines of code.

5.4.4.3 *Ultrasonic Sensor*

Write a function calcBlocks(x) which takes a float x as a parameter. Your program must measure distance measurement to the nearest obstacle using the Ultrasonic Sensor, and then pass the value to the function, which will then calculate the number of blocks to this obstacle, and then move forward that many blocks.
Hint: Each block is 20cm x 20cm big

5.4.4.4 *Camera*

Write a function colourMove(x) which takes a string parameter. Your program must use the camera to detect the colour of the current block, and then pass this colour to the function to decide which action to take. If the image is red, move forward one block, if the image is blue, turn right 2 steps, and if the image is green, reverse one block.

6. ANALYSIS OF RESULTS

Dependent Variables: ARCS

Independent Variables: Educational Tool ('X' or Robot)

6.1 Analysis of Likert Scale Questions

Table 1 below shows the first state of analysis done on the Likert Scale questions and the different components of motivation. A Mean and Standard Deviation value is given for each of the questions (N=7), as well as for the ARCS components themselves. In addition to the mean and standard deviation, the four components also have a *p*-value which was calculated using a two-sample t-test assuming equal or unequal variances, which was checked using an F-test. All components had equal variance except the Relevance component.

An ANOVA (Analysis of Variance) was done between the two sets of average scores for each component, pre and post. ANOVA ($F(1,6) = 22.4$, $p = <0.005$). A t-test of the same thing ($t(5)=-4.73$, two-tail $p = 0.005$).

An ANOVA was then run to test the effects of the independent variable on each of the dependent variables. Attention ($F(1,12) = 20.22$, $p < 0.005$). Relevance ($F(1,12) = 4.13$, $p < 0.5$). Confidence ($F(1,12) = 5.84$, $p < 0.05$). Satisfaction ($F(1,12) = 18.69$, $p < 0.005$).

Figure 9 below shows a more visual distribution of the scores given to each component in the pre and post questionnaire. The components are labelled with their first letter of their name, A/R/C/S followed by either an 'a' or 'b' for before and after the tasks were completed with the robot.

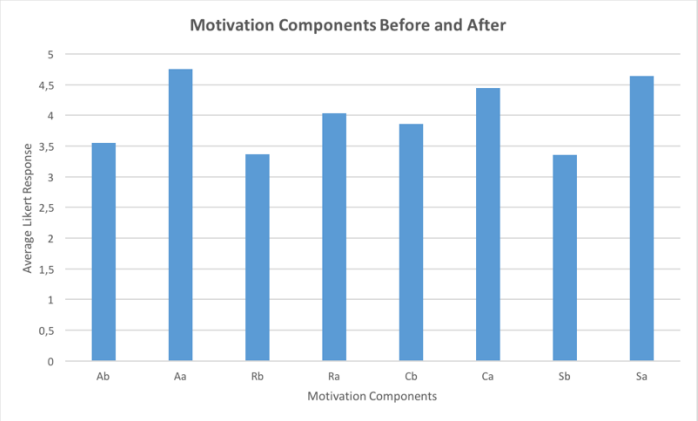

Figure 9: Bar Chart displaying the Scores for the four componenets of Motivation, before and after the testing.

Table 1. ARCS Survey Results

	Pre		Post		t-test
	M	SD	M	SD	*p*
Attention	**3.55**	**0.63**	**4.75**	**0.32**	
There is something interesting that got my attention about learning how to program with the robot.	3.00	0.82	4.86	0.38	
The way the robot was used to teach programming helped keep my attention.	4.14	0.69	5.00	0.00	
Learning how to program with the robot stimulated my curiosity.	3.86	0.90	5.00	0.00	
The work that we did learning how to program with the robot held my attention.	3.14	1.07	4.86	0.38	<0.005
The variety of uses of the robot helped keep my attention.	3.86	1.35	4.57	0.53	
Using the robot to learn how to program made me feel rewarded for my effort.	3.71	1.11	4.71	0.76	
It was a pleasure to work with the robot to learn how to program.	3.43	0.53	4.71	0.49	
I enjoyed using the robot to learn how to program so much that I would like to know more about it.	3.29	1.11	4.29	0.95	
Relevance	**3.37**	**0.79**	**4.03**	**0.34**	
Learning how to program with the robot will be (was) useful for me to learn how to program effectively.	2.86	0.90	3.86	0.90	
It is clear to me how using robots is related to learning how to program.	3.86	0.38	4.14	0.69	
There are sufficient stories, materials, and examples that showed me how the robot could be important to some people who are learning how to program.	3.57	1.51	3.86	1.21	<0.5
I see the usefulness of using robots to learn how to program.	4.00	0.82	4.71	0.49	
I could relate learning how to program with the robot to things I have seen, done, or thought about in my own life.	2.57	1.27	3.57	0.53	
Confidence	**3.86**	**0.52**	**4.45**	**0.36**	
I could understand quite a bit of the material involved with learning how to program with the robot.	4.43	0.53	4.43	0.53	
The programming exercises with the robot were too difficult.	4.00	1.00	4.57	0.79	
Learning how to program with the robot was easy and it was easy to remember the important points about using it.	3.29	0.76	4.43	0.53	<0.5
Learning how to program with the robot was so abstract that it was hard to keep my attention on it.	4.00	0.00	4.29	1.11	
Given that these tasks teach me how to program with robots, I am confident that I will complete the tasks successfully.	3.86	0.38	4.29	0.49	
Learning how to program with the robot was more difficult to understand than I would like for it to be.	3.71	0.76	4.71	0.49	
Using the robots to learn how to program is (was) intimidating to me.	3.71	1.25	4.43	0.53	
Satisfaction	**3.36**	**0.69**	**4.64**	**0.38**	
The amount of repetition using the robot to learn how to program caused me to be bored.	2.71	1.95	4.29	0.76	<0.005
I enjoyed learning how to program with the robot.	4.00	0.58	5.00	0.00	

7. DISCUSSION

The results found in the analysis of the motivational survey, consisting of the four ACRS components, show quite a significant difference. There is a clear increase in overall motivation with a p value of < 0.005. When the components were analyzed separately it was found that there was a more significant increase in the Attention and Satisfaction components, both having p values of < 0.005 whilst the Relevance and Confidence components had p values of <0.5. These results came as expected, from previous papers as well as personal experience from creating this software, the hardware component of the robot does appear to affect the confidence of the participants, because the hardware isn't constant and perfect and sometimes acts up, it occasionally makes you think your code is acting incorrectly while in the meantime it's the hardware. This is an important component as students, especially when they are new to the field and are starting to learn the basic concepts of programming, have to have confidence in the tools they are using. The relevance component has also come up before as you can't expect everyone to be interested in robotics, the ones that aren't that amazed by the idea would see them as irrelevant in learning to program, they might prefer to learn whilst creating applications for a smartphone. However, the system does attempt to abstract the hardware into a high-level language to help the focus of the assignments be on the code rather than the hardware, allowing the robot to have more visual feedback from the code than other basic tools such as an IDE. A possible solution to the problem with confidence would be to spend money on higher quality parts or have one designed especially from scratch instead of combining different parts. The satisfaction component, aside from having a clear increase in survey results, could be clearly seen during most of the testing with the participants, only one or two didn't show high levels of excitement in their reactions when the robot moved.

Even though testing wasn't done with students from the class the system is designed for, the results from testing with expert users does provide enough confidence that the system works well and isn't worse than what is currently being used. Allowing us to answer the research question proposed at the start of the project with yes, a robot could be used to increase the motivation of students learning programming in the CS1 extended degree course or at the very least, do no harm. The addition of the web-app seems to have helped a lot by increasing the speed of the write-run-debug loop which Fagin found in his research to be the reason for his poor results [2]. As McGill suggested in her future work, a control group was added [1], and tested against, by comparing the motivation of the robot to something else. This seems to have helped analyze the motivation of the robot more effectively.

The robot does however require a bit of maintenance, because of the inaccuracies of the motors, the robot will have to be physically reset every so often as well as the power bank plugged in and charged. The robot also requires quite a lot of space as well as a perfectly smooth surface with little friction which proved to be an issue when trying to find space to setup the enclosure and run the tests.

8. ETHICAL ISSUES

Because of the fact that the project would make use of University students when performing its user tests, ethical clearance had to be applied for from the Faculty of Science Research Ethics Committee. After which a DSA 100 application had to be made to the Department of Student Affairs for clearance. This was done in advance and there were no issues with the applications, ethical clearance was given, provided that participants signed an informed consent form before testing began.

9. LIMITATIONS

9.1 Hardware

Because of the problems that were previously mentioned about the robot not going perfectly straight, research was done into why, and consultations were held with the Engineering Department. It would appear that even once you have the motors running at the exact same speeds there are a few things which can still cause the robot to turn, these include unequal wheel diameters, uneven mass distribution and the misalignment of wheels. As well as the environmental effects of an uneven floor, wheel slippage, the robot bouncing against obstacles, fast turning etc. It was suggested that many of these problems could arise from the wheels and motors being fairly inexpensive.

UCT's campus has Wi-Fi boosters in almost every room, especially the Computer Science building and little to our knowledge before-hand, the infrared sensor picks up the Wi-Fi signals as an object in front of it, causing the infrared sensor not to work properly in certain areas and detecting obstacles when facing in the direction of a booster.

The number of available pins on the Raspberry Pi became a small challenge after the motors and all the sensors were plugged in there was almost no pins left at all for any additional features. The space on the Chassis was also limited and every time a new feature was added and tested on the robot, everything had to be shuffled around while still trying to evenly distribute the weight across the robot.

9.2 Testing

9.2.1 Participants

We were unable to use CSC1010H extended degree students because of the cancellation of lectures and shut down of campuses throughout the weeks of scheduled testing. The use of testing with expert users does come with certain limitations in that the best we can get from the participants is usability improvements as well as a professional opinion whether the robot would be something that could help to motivate students and if so, possibly be tested further in the near future.

9.2.2 Environment

Testing took place in the Honours Lab, which is not a quiet and controlled environment at all times. There was no protection against external factors such as noise and possible distractions. Because testing had to take place in the Honours Lab, which wasn't planned, we didn't have the setup that we wanted for the robot, a smaller enclosure/grid had to be improvised by putting two tables together and mounting the camera on the wall instead of the ceiling as planned.

9.3 External Factors

At the time of testing, UCT's campus wasn't a stable environment in itself and participants might not have been comfortable being there or in a hurry to leave.

10. FUTURE WORK

Future work could see further testing with the students from the CSC1010H course in order to make sure the system is built to their requirements and to make sure it's something they think they would be interested in learning with. After which the system could be integrated into the CSC1010H curriculum, for a semester or even a year, after which more analysis of its use could be done

and comparison to the previous year's marks on assignments could be made much like how Fagin analyzed his system [2][3]. A more validated study would be made possible by having students only use the robots from the beginning of the course, having them learning the concepts with the system, for the first time and preventing any prior knowledge or influences of other software on the results.

However, because of the low levels of confidence already felt by the expert users, it might be better to keep the current curriculum and to make the system available to the students to play with in their extra time, to practice the concepts they have learnt with example assignments.

11. ACKNOWLEDGEMENTS

I'd like to acknowledge my team members, Jeremy Coupland and Muhummad Patel, our Supervisor Gary Stewart, second reader Maria Keet, as well as the UCT Computer Science Department and all of our expert users that gave their time to help evaluate our system.

12. REFERENCES

[1] McGill, M. M. 2012. Learning to program with personal robots: Influences on student motivation. ACM Transactions on Computing Education (TOCE), 12, 1 (Jan. 2012), 4.

[2] B. S. Fagin and L. Merkle. Quantitative analysis of the e_ects of robots on introductory computer science education. Journal on Educational Resources in Computing (JERIC), 2(4):2, 2002.

[3] Fagin, B. and Merkle, L. 2003. Measuring the effectiveness of robots in teaching computer science. ACM SIGCSE Bulletin, 35, 1 (Jan. 2003), 307-311.

[4] Alimisis, D. 2012. Robotics in education & education in robotics: Shifting focus from technology to pedagogy. In *Proceedings of the 3rd International Conference on Robotics in Education.*, 7-14.

[5] Malec, J. 2001. Some thoughts on robotics for education. In *2001 AAAI Spring Symposium on Robotics and Education*.

[6] Saad, A., Shuff, T., Loewen, G. and Burton, K. 2012. Supporting undergraduate computer science education using educational robots. In *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 343-344.

[7] Blank, D. 2006. Robots make computer science personal. Commun ACM, 49, 12 (Dec. 2006), 25-27.

[8] Yadin, A. 2011. Reducing the dropout rate in an introductory programming course. ACM inroads, 2, 4 (Apr. 2011), 71-76.

[9] Sanders, I. and Scholtz, T. 2012. First year students' understanding of the flow of control in recursive algorithms. African Journal of Research in Mathematics, Science and Technology Education, 16, 3 (Mar. 2012), 348-362.

[10] D. Xu, D. Blank, and D. Kumar. Games, robots, and robot games: complementary contexts for introductory computing education. In Proceedings of the 3rd international conference on Game development in computer science education, pages 66{70. ACM, 2008.

[11] Cooper, S., Dann, W. and Pausch, R. 2000. Alice: a 3-D tool for introductory programming concepts. Journal of Computing Sciences in Colleges, 15, 5 (May. 2000), 107-116.

[12] Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D. and Balch, T. 2009. Personalizing CS1 with robots. ACM SIGCSE Bulletin, 41, 1 (Jan. 2009), 433-437.

[13] Markham, S. A. and King, K. 2010. Using personal robots in CS1: experiences, outcomes, and attitudinal influences. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. ACM, 204-208.

[14] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J., Floreano, D. and Martinoli, A. 2009. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions.* IPCB: Instituto Politécnico de Castelo Branco, 59-65.

[15] McNamara, S., Cyr, M., Rogers, C. and Bratzel, B. 1999. LEGO brick sculptures and robotics in education. In *ASEE Proc.*

[16] Blank, D., Kumar, D., Meeden, L. and Yanco, H. 2004. Pyro: A python-based versatile programming environment for teaching robotics. Journal on Educational Resources in Computing (JERIC), 4, 3 (Mar. 2004), 3.

[17] Saad, A. and Kroutil, R. M. 2012. Hands-on learning of programming concepts using robotics for middle and high school students. In *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 361-362.

[18] Fagin, B. S., Merkle, L. D. and Eggers, T. W. 2001. Teaching computer science with robotics using Ada/Mindstorms 2.0. ACM SIGAda Ada Letters, 21, 4 (Apr. 2001), 73-78.