Educational robots for teaching programming to youths

Adam Bowes

Date

# Abstract

 **What you set out to do and why**

 **How you did it**

 **What you found**

Do this last

# Table of Contents

[Abstract 2](#_Toc413512945)

[Table of Contents 3](#_Toc413512946)

[Aims and Objectives 4](#_Toc413512947)

[Literature Review 5](#_Toc413512948)

[Methodology 6](#_Toc413512949)

[Evaluation 7](#_Toc413512950)

[Critical Reflection 8](#_Toc413512951)

[Appendices 9](#_Toc413512952)

# Aims and Objectives

Note - Not sure if this is needed or the format.

Robots have been used to further education and to increase engagement in a range of topics. For example, in secondary school mathematics robots have been used to demonstrate geometric transformations. Programming is a skill with which many have encountered difficulties understanding and persisting with, one cause is a lack of motivation and enthusiasm towards the topic. Another cause is that the interaction with the computer is limited to the computers screen preventing any physical or real world feedback. In this research I have created an interface to allow the use of Scratch, a user friendly programming language to control a Thymio-II robot to allow users to have their programs affect the real world.

In order to achieve this I have completed several objectives.

# Literature Review

# Methodology

## Project management

Agile/learn/extreme

## Software Development

-software development methodology  
-characteristics of the software and the computer environments available.  
-simplistic review of strengths and weaknesses

This project required a piece of software to be developed. This software took form as an interface allowing communications between the programming language Scratch 1.4 and the Thymio-II robot. To create this interface the incremental software development method was applied. Incremental development is a variation on the waterfall method which consists of multiple waterfalls with reviews between each allowing waterfall to be used on flexible projects.

This method was chosen after considering the characteristics of the software being built and the environment in which it was constructed. Developing the interface required overcoming unique challenges which were largely unique to this type of software, as a result some segments of the code needed to be rewritten and upgraded as new knowledge was gained. Incremental development allows for the software to be reviewed after each task is completed so that improvements to the code can added as tasks at suitable points in the development. Consistently improved code was important for this project as this meant the scope of the interface could expand without causing conflicts with obsolete code. Scratch 1.4 and the Thymio-II robot both needed to communicate with the interface. As neither components were built with compatibilities for the other the development process had to account for impassable obstacles which could result in fundamental aspects of the interface needing to be changed such as the programming language or the methods of communication. Incremental development suited this as its review process gives an opportunity to evaluate the current state of the program in addition allowing newly acquired knowledge to be applied to the situation.

The result of using this method meant that during initial development when experimentation took place there was little commitment to carrying on with a particular plan. Following this unsuccessful ideas could be discarded quickly and so that the core functionality of the interface was more robust. The incremental aspect of the development meant that the software was completed in discrete and complete chunks such that each iteration resulted in a working prototype with individually changing features. This discrete improvement meant that progress could be monitored easily by reviewing which features had been added or changed. In addition to this the scope of the interface was expand where appropriate during the brief review periods to include feature which previously could not be planned for as the relevant knowledge about the system was not known.

One issue with the method was that features were often completed on their own without consideration for future iterations. This caused large amounts of the interface to need to be rewritten, although this had the advantage of ensuring the program was built with recently acquired knowledge it also caused the progress of the project to slow down. With a more rigid method features could be anticipated and prepared for better. Another issue with the development method was that while it was useful for increasing the scope of the software and monitoring progress it wasn’t very useful for estimating the tasks remaining and the time they would take. As the software needed to be functional before any testing or studies could take place

Cms, (2005) *Selecting Developing Approach*. [Online] Available from: www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf [Accessed 18 October 2014].

# Implementation

# Evaluation

# Critical Reflection

# Appendices